Thomas L. Floyd

9ª edición

Fundamentos de sistemas digitales

www.librosite.net/floyd

PEARSON
Prentice Hall

FUNDAMENTOS
DE
SISTEMAS DIGITALES

FUNDAMENTOS DE SISTEMAS DIGITALES

Novena Edición

THOMAS L. FLOYD

Traducción

Vuelapluma

Revisión Técnica

Eduardo Barrera López de Turiso

Departamento de Sistemas Electrónicos y de Control

Universidad Politécnica de Madrid



Madrid ● México ● Santa Fe de Bogotá ● Buenos Aires ● Caracas ● Lima
Montevideo ● San Juan ● San José ● Santiago ● São Paulo ● White Plains ●

Datos de catalogación bibliográfica

FUNDAMENTOS DE SISTEMAS DIGITALES
Thomas L. Floyd
PEARSON EDUCACIÓN S.A., Madrid, 2006

ISBN 10: 84-8322-085-7
ISBN 13: 978-84-832-2720-6
Materia: Informática, 0004.4

Formato: 195 x 250 mm. Páginas: 1024

Ribera del Loira, 28
28042 Madrid

FUNDAMENTOS DE SISTEMAS DIGITALES
Thomas L. Floyd

ISBN 10: 84-8322-085-7
ISBN 13: 978-84-8322-085-6

Deposito Legal:
PRENTICE HALL es un sello editorial autorizado de PEARSON EDUCACIÓN S.A.

Authorized translation from the English language edition, entitled DIGITAL FUNDAMENTALS, 9TH Edition by FLOYD, THOMAS L., published by Pearson Education Inc, publishing as Prentice Hall, Copyright © 2006

EQUIPO EDITORIAL
Editor: Miguel Martín-Romo
Técnico editorial: Marta Caicoya

EQUIPO DE PRODUCCIÓN:
Director: José A. Clares
Técnico: María Alvear

Diseño de Cubierta: Equipo de diseño de Pearson Educación S.A.

Impreso por:

IMPRESO EN ESPAÑA - PRINTED IN SPAIN

Este libro ha sido impreso con papel y tintas ecológicos

CONTENIDO

■ *Los temas marcados con este símbolo pueden considerarse opcionales.*

PREFACIO

Bienvenido a *Fundamentos de sistemas digitales. Novena edición.* Unos conocimientos sólidos sobre los fundamentos básicos de la tecnología digital son imprescindibles para cualquiera que desee desarrollar una carrera en esta excitante industria. Este texto se ha organizado cuidadosamente para incluir información actualizada de temas que pueden cubrirse por completo, utilizarse en formato condensado, u omitirse dependiendo del enfoque del curso.

Los temas tratados en el texto se cubren con el mismo formato claro, directo y cuidadosamente ilustrado que se ha empleado en las ediciones anteriores. Muchos temas se han reforzado o mejorado y pueden encontrarse numerosas mejoras a lo largo del libro.

Probablemente, encontrará más temas de los que se pueden cubrir en un curso cuatrimestral. Este amplio rango de temas proporciona la flexibilidad para diseñar una amplia variedad de cursos. Por ejemplo, algunos de los temas orientados al diseño o a las aplicaciones de los sistemas pueden no ser apropiados en algunos cursos. Otros cursos pueden no cubrir la lógica programable, mientras que otros pueden no disponer del tiempo necesario para tratar temas como las computadoras, microprocesadores o el procesamiento digital de la señal. También, en algunos cursos puede no ser necesario entrar en los detalles de la circuitería interna de los chips. Estos y otros temas se pueden omitir o verse por encima sin que los temas fundamentales se vean afectados. Disponer de conocimientos sobre los circuitos de transistores no es un prerrequisito para este libro de texto, aunque la tecnología de circuitos integrados se cubre en un "capítulo flotante", que es opcional.

El texto tiene una organización modular que permite incluir o excluir varios temas sin que tenga repercusión sobre el resto de los temas incluidos en un determinado curso. Dado que la lógica programable está adquiriendo cada vez más importancia, se ha dedicado un capítulo completo al tema (Capítulo 11), incluyendo el estudio de los dispositivos PAL, GAL, CPLD y FPGA, así como de dispositivos específicos de Altera y Xilinx. También se ha incluido una introducción de carácter general al software de los dispositivos lógicos programables.

Nuevo en esta edición

- Código de detección y corrección de errores Hamming
- Sumadores con acarreo anticipado
- Una breve introducción a VHDL
- Información ampliada y mejorada sobre instrumentos de prueba
- Información ampliada y reorganizada sobre los dispositivos lógicos programables y su software.
- Información mejorada sobre la localización de averías
- Nuevo enfoque en las secciones Aplicación a los sistemas digitales

Características

- Notas intercaladas en el texto que proporcionan información en un formato resumido.
- Las palabras clave se enumeran al principio de cada capítulo. Dentro del capítulo, estas palabras clave se resaltan en negrita y cursiva. Cada palabra clave se define al final del capítulo, así como al final del libro en un extenso glosario.

- El Capítulo 14 se ha diseñado como un capítulo “flotante” para proporcionar información opcional sobre la tecnología de circuitos integrados (circuitería interna del chip), que se puede estudiar en cualquier momento a lo largo del curso.
- Al principio de cada capítulo se enumeran los objetivos y se hace una breve introducción.
- Se incluye al principio de cada sección una introducción y los objetivos de la misma.
- Al final de cada sección se plantean ejercicios y cuestiones de repaso.
- Se incluye un problema relacionado en cada ejemplo resuelto.
- Se han intercalado Notas Informáticas a lo largo del texto que proporcionan información interesante sobre la tecnología informática relacionada con la cuestión que se está estudiando.
- Consejos prácticos intercalados proporcionan información útil y práctica.
- Las secciones Aplicación a los sistemas digitales se incluyen al final de muchos de los capítulos y exponen aplicaciones interesantes y prácticas de los fundamentos de los sistemas lógicos.
- Resúmenes al final de cada capítulo.
- Autotest con múltiples respuestas al final de cada capítulo.
- Conjuntos de problemas organizados por secciones al final de cada capítulo, incluyendo problemas básicos, de localización de averías, de aplicaciones de sistemas y de diseños especiales.
- Se cubre el uso y aplicación de instrumentos de prueba, como el osciloscopio, el analizador lógico, el generador de funciones y los multímetros digitales (DMM).
- El Capítulo 12 proporciona una introducción a las computadoras.
- El Capítulo 13 presenta el procesamiento digital de la señal, incluyendo la conversión analógica-digital y la conversión digital-analógica.
- Al principio del Capítulo 1 se presentan conceptos sobre la lógica programable.
- Se presentan a lo largo del texto circuitos integrados específicos que implementan una función determinada.
- El Capítulo 11 aborda los dispositivos PAL, GAL y FPGA, así como una exposición de carácter general sobre la programación de dispositivos PLD.
- En el Capítulo 11 se introduce la lógica de exploración de contorno asociada con los dispositivos programables.
- Además de la técnica de exploración de contorno, el tema de la localización de averías incluye otros métodos para probar los dispositivos programables, como las pruebas tradicionales y las camas de pinchos.
- Para aquellos que deseen incluir una introducción a la programación con ABEL, pueden encontrar información en el sitio web **www.librosite.net/floyd**.

Otros recursos para el estudiante

- *Experiments in Digital Fundamentals* de David M. Buchla es un manual de laboratorio. Las soluciones de este manual están disponibles en el manual del profesor *Instructor's Resoruce Manual.*

Recursos para el profesor

- *Sitio web* **www.librosite.net/floyd**. Este sitio web ofrece al profesor la posibilidad de publicar su plan de estudios en línea con nuestro programa Syllabus Manager™. Se trata de una excelente solución para la enseñanza a distancia, autodidacta o asistida por computadora.

- *Instructor's Resource Manual*. Este manual incluye las soluciones a los problemas planteados en los capítulos, las soluciones a las secciones de Aplicación a los sistemas digitales y los resultados de laboratorio para el manual de David M. Buchla (impreso y en línea)
- *Test Item File*. Esta edición incorpora más de 900 cuestiones.
- *TestGen*.® Es una versión electrónica de *Tests Item File*, que permite a los profesores personalizar los exámenes para cada curso.

Características de los capítulos

Introducción del capítulo Las dos primeras páginas de cada capítulo tienen el formato que se indica en la Figura P.1. La página de la izquierda contiene la lista de las secciones y la lista de los objetivos del capítulo. En la página de la derecha se presenta la introducción, una lista de los dispositivos específicos que se verán en el capítulo (cada nuevo dispositivo se indica mediante el logotipo de un circuito integrado en el lugar donde se introduce), una breve descripción de la aplicación a los sistemas digitales que se verá en el capítulo y una lista de palabras clave.

Introducción de la sección Cada sección del capítulo comienza con una breve introducción, que proporciona una visión general y una lista de los objetivos de la misma. En la Figura P.2 se muestra un ejemplo.

Revisión de la sección Cada sección termina con una revisión, en la que se incluyen preguntas o ejercicios sobre los principales conceptos presentados, como se muestra en la Figura P.2. Las respuestas a estos ejercicios se encuentran al final de cada capítulo.

3

PUERTAS LÓGICAS

CONTENIDO DEL CAPÍTULO

3.1 El inversor
3.2 La puerta AND
3.3 La puerta OR
3.4 La puerta NAND
3.5 La puerta NOR
3.6 Puertas OR–exclusiva y NOR–exclusiva
3.7 Lógica programable
3.8 Lógica de función fija
3.9 Localización de averías

OBJETIVOS DEL CAPÍTULO

- Describir el funcionamiento del inversor y de las puertas AND y OR.
- Describir el funcionamiento de las puertas NAND y NOR.
- Expresar las operaciones de las puertas NOT, AND, OR, NAND y NOR mediante el Álgebra de Boole.
- Describir el funcionamiento de las puertas OR–exclusiva y NOR–exclusiva.
- Reconocer y utilizar los símbolos distintivos y los símbolos rectangulares de las puertas lógicas según el estándar ANSI/IEEE 91–1984.
- Elaborar los diagramas de tiempos que muestran las relaciones de tiempo de las entradas y las salidas de las diferentes puertas lógicas.
- Establecer las comparaciones básicas entre las principales tecnologías de circuitos integrados: TTL y CMOS.
- Explicar las diferencias entre las series de las familias TTL y CMOS.
- Definir, para las puertas lógicas, los siguientes parámetros: *tiempo de retardo de propagación, disipación de potencia, producto velocidad–potencia y fan–out*.
- Enumerar circuitos integrados de función fija que contengan varias puertas lógicas.
- Utilizar cada puerta lógica en aplicaciones sencillas.
- Localización de averías en las puertas lógicas debidas a circuitos abiertos o cortocircuitos, utilizando el pulsador y la sonda lógica o el osciloscopio.

PALABRAS CLAVE

- Inversor
- Tabla de verdad
- Diagrama de tiempos
- Álgebra booleana
- Complemento
- Puerta AND
- Habilitar
- Puerta OR
- Puerta NAND
- Puerta NOR
- Puerta OR–exclusiva
- Puerta NOR–exclusiva
- Matriz AND
- Fusible
- Antifusible
- EPROM
- EEPROM
- SRAM
- Dispositivo objetivo
- JTAG
- CMOS
- TTL
- Tiempo de retardo de propagación
- Fan–out
- Carga unidad

INTRODUCCIÓN

Este capítulo hace énfasis en el funcionamiento lógico, las aplicaciones y la localización de averías de las puertas lógicas. Se cubre la relación entre las formas de onda de entrada y de salida de una puerta utilizando los diagramas de tiempos.

Los símbolos lógicos que se usan para representar las puertas lógicas están de acuerdo con el estándar ANSI/IEEE 91–1984. Este estándar ha sido adoptado por la industria privada, y la industria militar lo utiliza para su documentación interna así como para sus publicaciones.

En este capítulo se aborda tanto la lógica programable como la lógica de función fija. Puesto que en todas las aplicaciones se usan los circuitos integrados (CI), generalmente, la función lógica de un dispositivo es más importante para el técnico que los detalles de operación del circuito en el nivel de componentes en el interior del CI. Por tanto, la cobertura detallada de los dispositivos en el nivel de componente puede tratarse como un tema opcional. Para aquéllos que lo necesiten y tengan tiempo, en el Capítulo 14 se cubren las tecnologías de los circuitos integrados digitales, haciéndose referencia a partes del mismo a lo largo del texto. *Sugerencia*: repase la Sección 1.3 antes de comenzar con este capítulo.

DISPOSITIVOS LÓGICOS DE FUNCIÓN FIJA

(SERIES CMOS Y TTL)

74XX00	74XX02	74XX04
74XX08	74XX10	74XX11
74XX20	74XX21	74XX27
74XX30	74XX32	74XX86
74XX266		

FIGURA P.1 Introducción del capítulo.

Ejercicios de revisión al final de cada sección.

Párrafo de introducción y lista de los objetivos de la sección al comienzo de la misma.

Las Notas Informáticas se encuentran a lo largo del texto

332 ■ FUNCIONES DE LA LÓGICA COMBINACIONAL

REVISIÓN DE LA SECCIÓN 6.1

1. Determinar la suma (?) y el acarreo de salida (C_{out}) de un semi-sumador para cada uno de los siguientes grupos de bits de entrada:
(a) 01 **(b)** 00 **(c)** 10 **(d)** 11

2. Un sumador completo tiene C_{in} = 1. ¿Cuánto vale la suma (Σ) y el acarreo de salida (C_{out}) cuando $A = 1$ y $B = 1$?

6.2 SUMADORES BINARIOS EN PARALELO

Para formar un sumador binario en paralelo se conectan dos o más sumadores completos. En esta sección aprenderemos los principios básicos de este tipo de sumador, de manera que podamos entender todas las funciones necesarias de entrada y salida cuando se trabaja con este tipo de dispositivos.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Utilizar sumadores completos para implementar un sumador binario en paralelo. ■ Explicar el proceso de adición en un sumador binario en paralelo. ■ Emplear la tabla de verdad para un sumador en paralelo de 4 bits. ■ Utilizar dos dispositivos 74LS283 para sumar dos números binarios de 4 bits. ■ Ampliar el sumador de 4 bits para poder realizar adiciones de 8 bits o 16 bits.

Como se ha visto en la Sección 6.1, un único sumador completo es capaz de sumar dos números binarios de 1 bit y un acarreo de entrada. Para sumar números binarios de más de un bit, se tienen que utilizar sumadores completos adicionales. Cuando se suman dos números binarios, cada columna genera un bit de suma y un 1 ó 0, correspondiente al bit de acarreo, que se añade a la columna inmediata de la izquierda, como se muestra a continuación con dos números de 2 bits.

Bit de acarreo de la columna de la derecha

```
 1
 11
+01
100
```

En este caso, el bit de acarreo de la segunda columna se convierte en un bit de suma.

Para sumar dos números binarios, se necesita un sumador completo por cada bit que tengan los números que se quieren sumar. Así, para números de dos bits se necesitan dos sumadores, para números de cuatro bits hacen falta cuatro sumadores, y así sucesivamente. La salida de acarreo de cada sumador se conecta a la entrada de acarreo del sumador de orden inmediatamente superior, como se muestra en la Figura 6.7 para un sumador de 2 bits. Téngase en cuenta que se puede usar un semi-sumador para la posición menos significativa, o bien se puede poner a 0 (masa) la entrada de acarreo de un sumador completo, ya que no existe entrada de acarreo en la posición del bit menos significativo.

En la Figura 6.7 los bits menos significativo (LSB) de los dos números se representan como A_1 y B_1. Los siguientes bits de orden superior se representan como A_2 y B_2. Los tres bits de suma son Σ_1, Σ_2 y Σ_3. Observe

NOTAS INFORMÁTICAS

Las computadoras realizan la operación de suma con dos números a un tiempo, denominados *operandos*. El *operando fuente* es un número que se añade a un número existente denominado *operando de destino*, que es el que se almacena en un registro de la UAL, tal como el acumulador. A continuación, la suma de los dos números se almacena de nuevo en el acumulador. La adición se realiza con números enteros o números en coma flotante utilizando, respectivamente, las instrucciones ADD o FADD.

FIGURA P.2 Introducción y revisión de una sección.

Ejemplos resueltos y problemas relacionados Numerosos ejemplos resueltos ayudan a ilustrar y clarificar los conceptos básicos o procedimientos específicos. Cada ejemplo concluye con un problema relacionado que le refuerza o amplia, que requieren que el estudiante resuelva siguiendo pasos similares a los seguidos en el ejemplo. En la Figura P.3 se muestra una página con un ejemplo resuelto típico y un problema relacionado.

Sección de localización de averías Muchos capítulos incluyen una sección dedicada a la localización de averías, que hace referencia a los temas cubiertos en el capítulo y que se centra en las técnicas de localización de averías y el uso de instrumentos de prueba. En la Figura P.4 se muestra una parte de una sección típica sobre la localización de averías.

Aplicación a los sistemas digitales La última sección de la mayor parte de los capítulos presenta una aplicación práctica sobre los conceptos y dispositivos cubiertos en el capítulo. Cada una de estas secciones presenta un sistema del mundo real, en el que se implementan las etapas de análisis, diseño y localización de averías utilizando los procedimientos vistos en el capítulo. Algunas de las aplicaciones a sistemas están limitadas a un único capítulo, y otras se extienden a lo largo de dos o más. Las aplicaciones a los sistemas digitales y sus capítulos asociados son las siguientes:

126 ■ PUERTAS LÓGICAS

EJEMPLO 3.1

Al inversor de la Figura 3.4 se le aplica una señal. Determinar la forma de onda de salida correspondiente a la entrada y dibujar el diagrama de tiempos. De acuerdo con el emplazamiento del círculo ¿cuál es el estado activo de salida?

FIGURA 3.4

Solución La forma de onda de salida es exactamente la opuesta a la de entrada (es la entrada invertida), como se muestra en la Figura 3.5, que es el cronograma básico. El estado activo o verdadero de salida es 0.

FIGURA 3.5

Problema relacionado* Si el inversor tiene el indicador negativo (círculo) en la entrada en lugar de en la salida, ¿cómo afecta esto al diagrama de tiempos?

* Las respuestas se encuentran al final del capítulo.

Cada ejemplo queda delimitado mediante un recuadro

Cada ejemplo contiene un problema relacionado con el mismo.

▲ *El álgebra booleana utiliza variables y operadores para describir un circuito lógico.*

encima de la letra. Una variable puede tomar uno de dos valores, 1 ó 0. Si una variable dada es 1, su complemento es 0, y viceversa.

El modo de operación de un inversor (circuito NOT) puede expresarse del siguiente modo: si la variable de entrada se designa por A y la variable de salida por X, entonces

$$X = \overline{A}$$

Esta expresión establece que la salida es el complemento de la entrada, de modo que si $A = 0$, entonces $X = 1$, y si $A = 1$, entonces $X = 0$. La Figura 3.6 ilustra esto. La variable complementada $\overline{A}$ se lee "A barra" o "no A".

FIGURA 3.6 El inversor complementa una variable de entrada.

Aplicación

La Figura 3.7 muestra un circuito que genera el complemento a 1 de un número binario de 8 bits. Los bits del número binario se aplican a las entradas del inversor y el complemento a 1 se obtiene en las salidas.

FIGURA P.3 Un ejemplo y un problema relacionado.

- Sistema de control y recuento de pastillas: Capítulo 1.
- Display digital: Capítulos 4 y 11.
- Sistema de control de un tanque de almacenamiento: Capítulo 5.
- Sistema de control de semáforos: Capítulos 6, 7 y 8.
- Sistema de seguridad: Capítulos 9 y 10.

Las aplicaciones a los sistemas digitales pueden tratarse como secciones opcionales, ya que su omisión no afecta al resto del material incluido en el texto. La Figura P.5 muestra una parte de una sección de "Aplicación a los sistemas digitales".

Fin del capítulo Al final de cada capítulo se incluye la siguiente información:

- Resumen
- Glosario de las palabras clave
- Autotest

3.9 LOCALIZACIÓN DE AVERÍAS

La localización de averías es el proceso de reconocer, aislar y corregir un fallo en un sistema o circuito. Para poder localizar las averías de forma efectiva, debe entender cómo se supone que trabaja el circuito o sistema y debe estar en disposición de reconocer un funcionamiento incorrecto. Por ejemplo, para determinar si una puerta lógica tiene un fallo, debe saber cuál debe ser la salida para unas entradas dadas.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Comprobar la existencia de entradas y salidas abiertas internamente en las puertas lógicas de los CI. ■ Reconocer los efectos de una entrada o una salida del CI cortocircuitada. ■ Detectar en una tarjeta de circuito impreso la existencia de fallos externos. ■ Localizar las averías en un sencillo contador de frecuencia utilizando un osciloscopio.

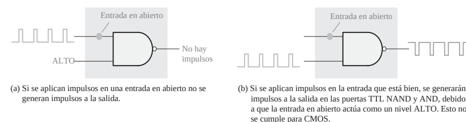
Fallos internos en las puertas lógicas de los CI

Los circuitos abiertos y los cortocircuitos son los fallos más comunes en las puertas internas del CI. Se pueden producir tanto en las entradas como en la salida de una puerta contenida en el encapsulado del CI. *Antes de intentar solucionar cualquier avería, compruebe que la alimentación continua y la masa son correctas.*

Efectos de una entrada que se encuentra en circuito abierto internamente. Un circuito abierto interno es el resultado de un componente en circuito abierto o de una ruptura en la conexión entre el chip y el pin del encapsulado. Una entrada en circuito abierto impide que una señal de impulsos en esta entrada dé lugar a una salida, como se muestra en la Figura 3.67(a) para la puerta NAND de 2 entradas. Una entrada TTL en abierto actúa como un nivel ALTO, por lo que los impulsos aplicados a la entrada que está en buen estado pasan a través de la puerta NAND hasta la salida, como se muestra en la Figura 3.67(b).

Condiciones para probar las puertas. Al probar una puerta NAND o una puerta AND, debe asegurarse siempre de que las entradas a las que no se aplican impulsos se encuentren a nivel ALTO, para activar la puerta. Cuando pruebe una puerta NOR o una puerta OR, debe asegurarse siempre de que las entradas a las que no se aplican impulsos se encuentran a nivel BAJO. Cuando se prueba una puerta XOR o XNOR, el nivel de la entrada a la que no se aplican impulsos no importa, ya que los impulsos aplicados en la otra entrada forzarán a que las entradas se encuentren, alternativamente, en el mismo nivel o en niveles opuestos.

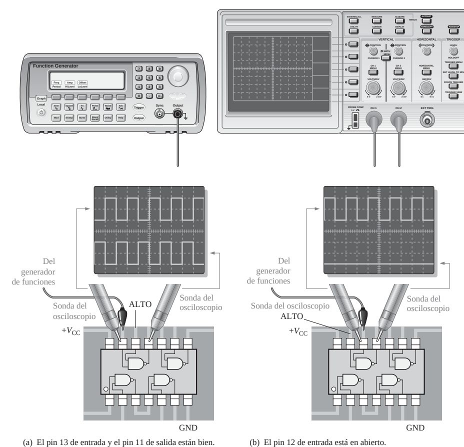
Localización de fallo: entrada en circuito abierto. La localización de este tipo de fallo es, en la mayoría de los casos, muy fácil utilizando un osciloscopio y un generador de funciones, como se muestra en la Figura 3.68, para el caso de una puerta NAND de 2 entradas. Al medir las señales digitales con un osciloscopio, emplee siempre el acoplamiento en continua.



(a) Si se aplican impulsos en una entrada en abierto no se generan impulsos a la salida.

(b) Si se aplican impulsos en la entrada que está bien, se generarán impulsos a la salida en las puertas TTL NAND y AND, debido a que la entrada en abierto actúa como un nivel ALTO. Esto no se cumple para CMOS.

FIGURA 3.67 Efecto de una entrada en circuito abierto en una puerta NAND.



(a) El pin 13 de entrada y el pin 11 de salida están bien.

(b) El pin 12 de entrada está en abierto.

FIGURA 3.68 Localización de averías en una puerta NAND con una entrada en circuito abierto.

El primer paso en la localización de averías de un CI, cuando se sospecha que está fallando, es asegurarse de que la tensión de alimentación continua (V_{CC}) y la masa están conectadas a los pines apropiados del CI. Después, se aplican impulsos continuos a una de las entradas de la puerta, asegurándose de que las otras entradas están a nivel ALTO (en el caso de una puerta NAND). En la Figura 3.68(a), se comienza a aplicar los impulsos en el pin 13, ya que se ha determinado que es una de las entradas de la puerta de la que se sospecha el fallo. Si en la salida correspondiente (en este caso el pin 11) se detecta un tren de impulsos, entonces el pin 13 de entrada no está en abierto. Consecuentemente, esto prueba también que la salida no está en abierto. A continuación, se aplica otro tren de impulsos a otra entrada de la puerta (pin 12), asegurándose de que la otra entrada está a nivel ALTO. En la salida (en el pin 11) no se detecta un tren de impulsos y la salida está a nivel BAJO, lo que indica que la entrada del pin 12 está en abierto, como se muestra en la Figura 3.68(b). Observe que la entrada en la que no se aplican impulsos debe estar a nivel ALTO en el caso de una puerta NAND o

FIGURA P.4 Páginas representativas de una sección típica dedicada a la localización de averías.

- ■ Un conjunto de problemas, que incluye algunas o todas las categorías siguientes: problemas básicos, problemas sobre localización de averías, problemas sobre aplicaciones a sistemas y problemas de diseño.
- ■ Respuestas a las revisiones de las secciones.
- ■ Respuestas a los problemas relacionados de los ejemplos.
- ■ Respuestas al autotest.

Al estudiante

¡La tecnología está de moda! Casi todo se está digitalizando o se digitalizará en un futuro próximo. Por ejemplo, los teléfonos móviles y otros medios de comunicación inalámbricos, la televisión, la radio, el control de procesos, la electrónica de automoción, la electrónica de consumo, las técnicas de posicionamiento global, los sistemas militares, por nombre sólo unas pocas aplicaciones, dependen enormemente de la electrónica digital.

Conocer en profundidad los fundamentos de la tecnología digital le preparará para poder acceder en un futuro a trabajos bien remunerados y de alta capacitación. Lo más importante que puede tratar de hacer el lector es comprender los fundamentos básicos. Habiéndolos dominado, tendrá en sus manos hacer lo que desee.

Además, la lógica programable está adquiriendo una importancia extraordinaria en el panorama tecnológico actual y ese es uno de los temas fundamentales cubiertos en el libro. Por supuesto, las habilidades necesarias para un diagnóstico eficiente también son altamente demandadas. El libro incluye, por ello, métodos de

386 ■ FUNCIONES DE LA LÓGICA COMBINACIONAL



APLICACIÓN A LOS SISTEMAS DIGITALES

En esta aplicación, vamos a comenzar a trabajar con el sistema de control de semáforos. En esta sección se establecen los requisitos del sistema, se desarrolla un diagrama de bloques, así como un diagrama de estados para ayudar a establecer la secuencia de funcionamiento. Diseñaremos la parte del sistema que involucra lógica combinacional y se propondrán los métodos de prueba. En los Capítulos 7 y 8 se tratarán los circuitos de lógica secuencial y de temporización del sistema.

Requerimientos generales del sistema

Se requiere un controlador digital para controlar un semáforo en la intersección de una calle de tráfico muy denso con una calle de tráfico moderado. La calle principal va a tener una luz verde durante un mínimo de 25 seg. o mientras no haya ningún vehículo en la calle perpendicular. Esta calle lateral tiene que tener la luz verde hasta que no circule ningún coche por ella, o durante un máximo de 25 seg. La luz ámbar de precaución tiene que durar 4 seg. en los cambios de luz verde a roja en ambas calles, principal y lateral. Estos requisitos se muestran en el diagrama de la Figura 6.65.

Desarrollo de un diagrama de bloques del sistema

A partir de los requisitos, se puede desarrollar un diagrama de bloques del sistema. En primer lugar, sabemos que el sistema tiene que controlar seis pares de luces diferentes. Estas son las luces roja, ámbar y verde para ambos sentidos, tanto en la calle principal como en la lateral. También sabemos que existe una entrada externa (además de la alimentación) que proviene de un sensor de vehículos situado en la calle lateral. En la Figura 6.66, puede ver un diagrama de bloques mínimo que ilustra estos requisitos.

A partir del diagrama de bloques mínimo vamos a ir entrando en los detalles. El sistema tiene cuatro estados, como se indica en la Figura 6.65, por lo que se necesita un circuito lógico para controlar la secuencia de estados (lógica secuencial). Además, se necesitan circuitos para generar los intervalos de tiempo adecuados de 25 seg. y 4 seg., que se requieren en el sistema y para generar una señal de reloj cíclica en el sistema (circuitos de temporización). Los intervalos de tiempo (largo y corto) y el sensor de vehículos son entradas de la lógica secuencial, dado que la secuenciación de estados es una función de estas variables. Se necesitan también circuitos lógicos para determinar cuál de los cuatro estados del sistema está activo en un determinado instante de tiempo, para así generar las salidas adecuadas en las luces (decodificación de estados y lógica de salida), y para iniciar los intervalos de tiempo largo y corto. Finalmente, se necesita un circuito de interfaz para convertir los niveles lógicos de la decodificación y del circuito de salida en las tensiones y corrientes requeridas para encender cada una de las luces. La Figura 6.67 representa un diagrama de bloques más detallado que muestra estos elementos esenciales.

APLICACIÓN A LOS SISTEMAS DIGITALES ■ 387

El diagrama de estados

Un diagrama de estados nos muestra gráficamente la secuencia de estados en un sistema y las condiciones de cada estado y de las transiciones entre cada uno de ellos. En realidad, la Figura 6.65 es, en cierta medida, un diagrama de estados, ya que muestra la secuencia de estados y las distintas condiciones.

Definición de las variables. Antes de poder desarrollar un diagrama de estados tradicional, es necesario definir las variables que determinan cómo pasa el sistema a través de los diferentes estados. A continuación se enumeran estas variables y sus símbolos:

- Presencia de vehículos en la calle lateral = V_s
- El temporizador de 25 s. (largo) está *activado* = T_L
- El temporizador de 4 s. (corto) está *activado* = T_S

El uso de variables complementadas indica la condición contraria. Por ejemplo, $\overline{V}_s$ indica que no hay ningún vehículo en la calle lateral; $\overline{T}_L$ indica que el temporizador de larga duración está desactivado y $\overline{T}_S$ indica que el temporizador de corta duración está desactivado.

Descripción del diagrama de estados. En la Figura 6.68 se muestra un diagrama de estados. Cada uno de los cuatro estados se etiqueta de acuerdo a la secuencia de 2 bits en código Gray, como se indica mediante los círculos. La flecha circular en cada estado indica que el sistema permanece en dicho estado bajo la condición definida por la variable o expresión asociada. Cada una de las flechas que van de un estado al siguiente indican un cambio de estado cuando se produce la condición definida por la variable o expresión asociada.

Primer estado El código Gray para este estado es 00. El semáforo de la calle principal está en verde y el de la calle lateral está en rojo. El sistema permanece en este estado al menos 25 segundos cuando el temporizador largo se encuentra activado o mientras que no haya ningún vehículo en la calle lateral $(T_L + \overline{V}_s)$. El sistema pasa al siguiente estado cuando el temporizador de 25 segundos está desactivado o cuando aparece algún vehículo en la calle secundaria $(\overline{T}_L V_s)$.

Segundo estado El código Gray para este estado e 01. El semáforo de la calle principal está en ámbar (precaución) y el de la calle lateral está en rojo. El sistema permanece en este estado durante 4 segundos mientras el temporizador corto está activado (T_S) y pasa al siguiente estado cuando este temporizador se desactiva $(\overline{T}_S)$.

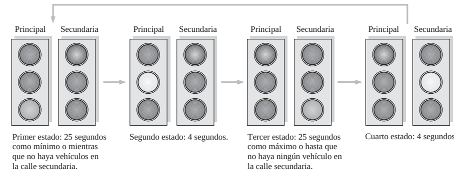


FIGURA 6.65 Requisitos para la secuencia de luces de los semáforos.

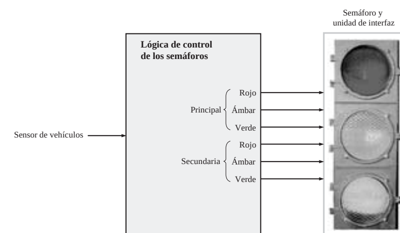


FIGURA 6.66 Diagrama de bloques mínimo del sistema.

FIGURA P.5 Páginas representativas de una sección típica dedicada a las aplicaciones de sistemas digitales.

prueba y localización de averías que van desde las pruebas tradicionales a las técnicas utilizadas en los procesos de fabricación, como la de cama de pinchos, la de sonda volante y la de exploración de contorno. Estos son algunos ejemplos de los conocimientos que podrá adquirir si se aplica con decisión al aprendizaje de los conceptos presentados.

Agradecimientos

Este innovador texto es el resultado de los esfuerzos y las habilidades de muchas personas. Creo que hemos conseguido lo que pretendíamos hacer, que era escribir un libro de texto de excelente calidad. En Prentice Hall, Kate Linsner y Rex Davidson han aportado su tiempo, talento y esfuerzo a lo largo de las muchas fases de este proyecto con la finalidad de obtener un libro como el que el lector tiene en sus manos. Lois Porter ha hecho un fantástico trabajo de edición del manuscrito. Ha sido capaz de desvelar los misterios de las marcas de este autor y con frecuencia sus prácticamente ilegibles anotaciones y, a partir de un desorden completo, extraer un manuscrito increíblemente organizado y estupendamente editado. También Jane Lopez ha hecho un estupendo trabajo con las imágenes. Otra persona que ha contribuido significativamente a este libro es Gary Snyder, quien ha proporcionado todos los archivos de los circuitos Multisim (en Multisim Versions 2001, 7 y 8, los cuales se incluyen en el sitio web de acompañamiento **www.librosite.net/floyd**). Quiero dar las gracias también a todas aquellas personas que de alguna manera, aunque sea de forma indirecta, han colaborado en este proyecto.

Para la revisión de éste y de todos los libros de texto, los autores dependemos de las inteligentes observaciones de los lectores y del equipo de revisión. Quiero dar mis más sinceras gracias a los siguientes revisores,

que han proporcionado multitud de sugerencias y han hecho una crítica enormemente constructiva: Bo Barry, Universidad de Carolina del Norte; Chuck McGlumphy, Belmont Tecnical College; y Amy Ray, Mitchell Community College.

Mi gratitud a David Buchla por sus esfuerzos a la hora de garantizar que el manual de laboratorio estuviera coordinado con el texto, así como por sus valiosas sugerencias. También quiero agradecer las sugerencias de Muhammed Arif Shabir en lo que respecta a los registros de desplazamiento.

Gracias a todos los miembros del equipo comercial de Prentice Hall, cuyo enorme trabajo ha ayudado a que mis libros estuvieran a disposición de un gran número de estudiantes en todo el mundo. Además, mi reconocimiento a todos vosotros, los profesores que habéis adoptado esta obra como libro de texto. Sin vosotros, esta obra no existiría. Espero que el lector encuentre en esta obra una valiosa herramienta de aprendizaje y un útil texto de referencia.

Tom Floyd

FUNDAMENTOS DE SISTEMAS DIGITALES

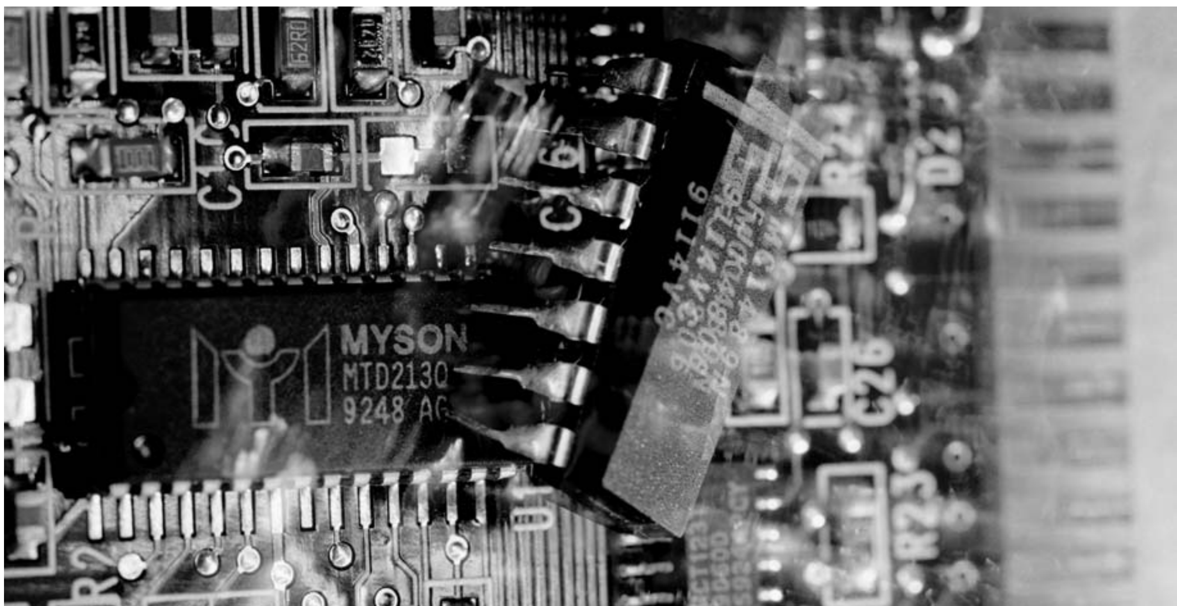
1

CONCEPTOS DIGITALES

CONTENIDO DEL CAPÍTULO

OBJETIVOS DEL CAPÍTULO

- Explicar las diferencias básicas entre las magnitudes digitales y analógicas.
- Mostrar cómo se usan los niveles de tensión para representar magnitudes digitales.
- Describir los diferentes parámetros de los trenes de impulsos, tales como el tiempo de subida, el tiempo de bajada, el ancho del impulso, la frecuencia, el período y el ciclo de trabajo.
- Explicar las operaciones lógicas básicas NOT, AND y OR
- Describir las funciones lógicas del comparador, el sumador, el convertidor de código, el codificador, decodificador, multiplexor, demultiplexor, contador y registro.



- Identificar los circuitos integrados digitales de función fija de acuerdo con su complejidad y el tipo de encapsulado.
- Identificar la numeración de los pines en los encapsulados de los circuitos integrados.
- Explicar qué es la lógica programable, especificando los distintos tipos y describiendo cómo se programan los PLD.
- Reconocer los instrumentos y comprender cómo se usan para medir y solucionar las averías en los sistemas y circuitos digitales.
- Mostrar en una aplicación práctica cómo se forma un sistema digital completo combinando las funciones básicas.

PALABRAS CLAVE

Las palabras clave están ordenadas de acuerdo con el orden de aparición a lo largo del capítulo.

- Analógico
- Digital
- Binario
- Bit
- Impulso
- Reloj
- Diagrama de tiempos
- Datos
- Serie
- Paralelo
- Lógica
- Entrada
- Salida
- Puerta
- NOT
- Inversor
- AND
- OR
- Circuito integrado (CI)
- SPLD
- CPLD
- FPGA
- Compilador
- Solución de averías

INTRODUCCIÓN

El término *digital* se deriva de la forma en que las computadoras realizan las operaciones contando dígitos. Durante muchos años, las aplicaciones de la electrónica digital se limitaron a los sistemas informáticos. Hoy día, la tecnología digital tiene aplicación en un amplio rango de áreas además de la informática. Aplicaciones como la televisión, los sistemas de comunicaciones, de radar, sistemas de navegación y guiado, sistemas militares, instrumentación médica, control de procesos industriales y electrónica de consumo, usan todos ellos técnicas digitales. A lo largo de los años, la tecnología digital ha progresado desde los circuitos de válvulas de vacío hasta los transistores discretos y los circuitos integrados, conteniendo algunos de ellos millones de transistores.

Este capítulo presenta la electrónica digital y proporciona una extensa introducción a muchos conceptos, componentes y herramientas muy importantes.

■■■ PRESENTACIÓN DE LA APLICACIÓN A LOS SISTEMAS DIGITALES

La última sección de la mayor parte de los capítulos de este libro se dedica a una aplicación, que resume los principales aspectos abordados en el capítulo. Cada sistema está diseñado para ilustrar, en cada capítulo, cómo pueden utilizarse la teoría y los dispositivos. A lo largo del libro, se presentan cinco sistemas diferentes, abarcando algunos de ellos dos o más capítulos.

Todos los sistemas se han simplificado para hacerlos manejables dentro del contexto del tema del capítulo. Aunque están basados en los requisitos reales del sistema, están diseñados para adecuarse al contenido del capítulo y no pretenden representar el método más eficiente o más moderno para dicha aplicación.

En este capítulo se presenta el primer sistema, que es un sistema de control industrial para contar y controlar los objetos de envasado que se encuentran sobre una cinta transportadora. Está diseñado para incorporar todas las funciones lógicas presentadas en el capítulo, de forma que se pueda ver cómo se utilizan y cómo interaccionan para conseguir un objetivo de utilidad.

1.1 MAGNITUDES ANALÓGICAS Y DIGITALES

Los circuitos electrónicos pueden dividirse en dos amplias categorías: digitales y analógicos. La electrónica digital utiliza magnitudes con valores discretos y la electrónica analógica emplea magnitudes con valores continuos. Aunque en este libro vamos a estudiar los fundamentos digitales, también debemos conocer los analógicos porque muchas aplicaciones requieren la utilización de ambos.

Al finalizar esta sección el lector deberá ser capaz de:

■ Definir el término *analógico*. ■ Definir el término *digital*. ■ Explicar las diferencias entre magnitudes digitales y analógicas. ■ Establecer las ventajas de digital frente a analógico. ■ Proporcionar ejemplos de cómo se utilizan en electrónica las magnitudes digitales y analógicas.

Una magnitud ***analógica**** es aquella que toma valores continuos. Una magnitud ***digital*** es aquella que toma un conjunto de valores discretos. La mayoría de las cosas que se pueden medir cuantitativamente aparecen en la naturaleza en forma analógica. Por ejemplo, la temperatura varía dentro de un rango continuo de valores. A lo largo de un día, la temperatura no varía por ejemplo entre 20ºC y 25ºC de forma instantánea, sino que alcanza todos los infinitos valores que hay en ese intervalo. Si dibujamos la gráfica de la temperatura de un día típico de verano , tendríamos una curva continua suave como la mostrada en Figura 1.1. Otros ejemplo de magnitudes analógicas son el tiempo, la presión, la distancia y el sonido.

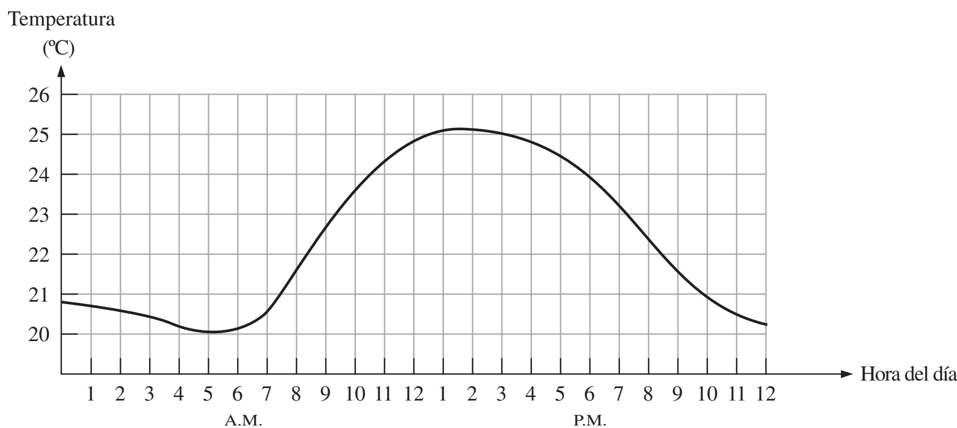


FIGURA 1.1 Gráfica de una magnitud analógica (temperatura en función del tiempo).

En lugar de hacer una gráfica de temperatura en un rango continuo, supongamos que simplemente medimos la temperatura cada hora. Lo que tenemos ahora son muestras que representan la temperatura en instantes discretos de tiempo (cada hora) a lo largo de un período de 24 horas, como se indica en la Figura 1.2. De esta forma, hemos convertido de forma efectiva una magnitud analógica a un formato que ahora puede digitalizarse, representando cada valor muestreado mediante un código digital. Es importante darse cuenta de que la Figura 1.2 no es la representación digital de la magnitud analógica.

La ventaja de las magnitudes digitales. En las aplicaciones de electrónica, la representación digital presenta ciertas ventajas sobre la representación analógica. La principal ventaja es que los datos digitales puede ser proce-

* Todos los términos en negrita son importantes y están definidos en el glosario al final del libro. Todos los términos en negrita y cursiva están incluidos en el glosario Palabras clave al final de cada capítulo.

sados y transmitidos de forma más fiable y eficiente que los datos analógicos. También, los datos digitales disfrutan de una ventaja importante cuando es necesario su almacenamiento. Por ejemplo, cuando la música se convierte a formato digital puede almacenarse de manera más compacta y reproducirse con mayor precisión y claridad de lo que es posible en formato analógico. El ruido (fluctuaciones de tensión no deseadas) no afecta a los datos digitales tanto como a las señales analógicas.

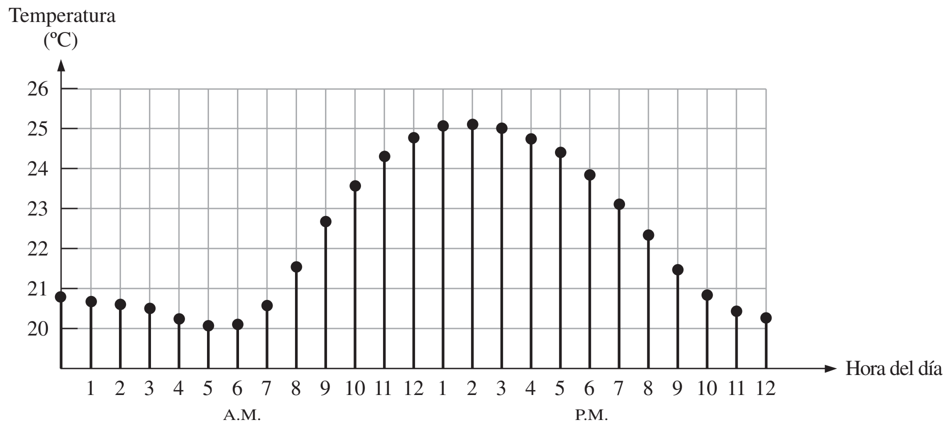


FIGURA 1.2 Representación de los valores muestreados (cuantificación) de la magnitud analógica de la Figura 1.1. Cada valor representado por un punto puede digitalizarse, representándolo como un código digital que consta de una serie de 1s y 0s.

Un sistema electrónico analógico

Un sistema de altavoz, que se emplea para amplificar el sonido de modo que pueda ser escuchado por una gran audiencia, es un ejemplo de una aplicación de electrónica digital. El diagrama básico de la Figura 1.3 ilustra cómo estas ondas sonoras, que son analógicas por naturaleza, son captadas por un micrófono y convertidas en una pequeña variación analógica de tensión denominada señal de audio. Esta tensión varía de forma continua a medida que el volumen y la frecuencia del sonido varían, y se aplica a la entrada de un amplificador lineal. La salida del amplificador, que es una reproducción amplificada de la tensión de entrada, se aplica al altavoz. El altavoz convierte de nuevo la señal de audio amplificada en ondas sonoras con un volumen mucho mayor que el sonido original captado por el micrófono.

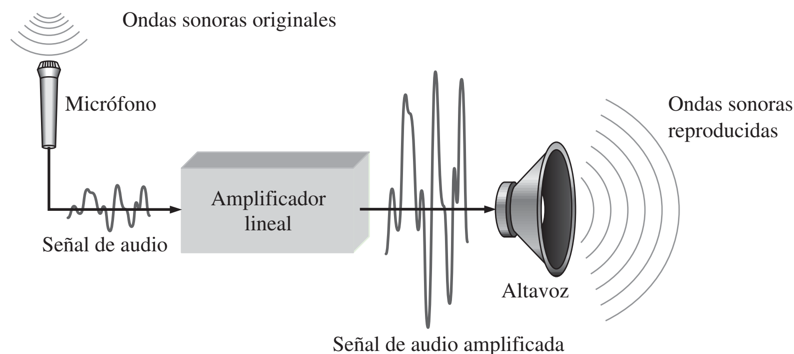


FIGURA 1.3 Sistema básico de altavoz.

Sistema que utiliza métodos digital y analógico

Un reproductor de CD es un ejemplo de un sistema en que se emplean tanto circuitos digitales como analógicos. El diagrama de bloques simplificado de la Figura 1.4 ilustra el principio básico. La música en formato digital se almacena en el CD. Un sistema óptico de diodos láser lee los datos digitales del disco cuando éste gira y los transfiere al **convertidor digital-analógico** (**DAC**, *Digital-to-Analog Converter*). El DAC transforma los datos digitales en una señal analógica que es una reproducción eléctrica de la música original. Esta señal se amplifica y se envía al altavoz para que podamos disfrutarla. Cuando la música original se grabó en el CD se utilizó el proceso inverso del descrito aquí, y que utilizaba un **convertidor analógico-digital** (**ADC**, *Analog-to-Digital Converter*).

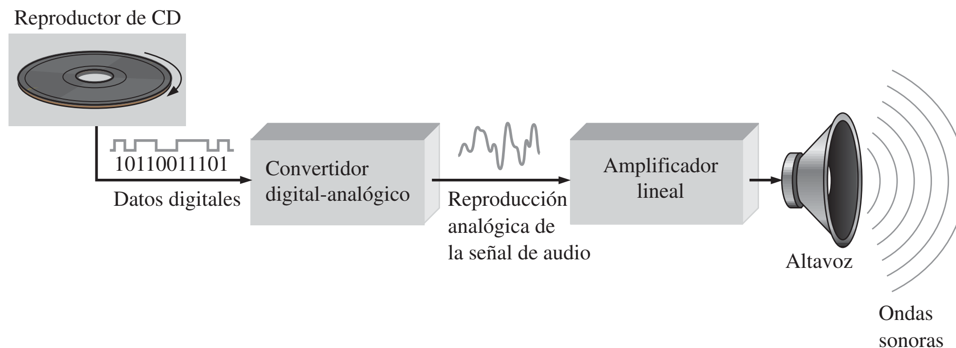


FIGURA 1.4 Esquema básico de bloques de un reproductor de CD. Sólo se muestra un canal.

REVISIÓN DE LA SECCIÓN 1.1

Las respuestas se encuentran al final del capítulo.

1. Definir *analógico.*
2. Definir *digital.*
3. Explicar la diferencia entre una magnitud digital y una magnitud analógica.
4. Proporcionar un ejemplo de un sistema que sea analógico y de otro que sea una combinación de digital y analógico. Nombrar un sistema que sea por completo digital.

1.2 DÍGITOS BINARIOS, NIVELES LÓGICOS Y FORMAS DE ONDA DIGITALES

La electrónica digital utiliza sistemas y circuitos en los que sólo existen dos estados posibles. Estos estados se representan mediante dos niveles de tensión diferentes: ALTO (HIGH) y BAJO (LOW). Estos dos estados también pueden representarse mediante niveles de corriente, bits y relieves en un CD o en un DVD, etc. En los sistemas digitales como las computadoras, las combinaciones de los dos estados, denominadas *códigos,* se emplean para representar números, símbolos, caracteres alfabéticos y otros tipos de datos. El sistema de numeración de dos estados se denomina *binario* y los dos dígitos que emplea son 0 y 1. Un dígito binario se denomina *bit.*

Al finalizar esta sección, el lector deberá ser capaz de:

■ Definir *binario*. ■ Definir *bit*. ■ Nombrar los bits en un sistema binario. ■ Explicar cómo se emplean los niveles de tensión para representar bits. ■ Explicar cómo un circuito digital interpreta los

niveles de tensión. ■ Describir las características generales de un impulso. ■ Determinar la amplitud, el tiempo de subida, el tiempo de bajada y el ancho de un impulso. ■ Identificar y describir las características de una forma de onda digital. ■ Determinar la amplitud, el período, la frecuencia y el ciclo de trabajo de una forma de onda digital. ■ Explicar qué es un diagrama de tiempos y establecer su finalidad. ■ Explicar la transferencia de datos paralelo y serie, y las ventajas y desventajas de cada una de ellas.



NOTAS INFORMÁTICAS

El concepto de computadora digital se remonta a Charles Babbage, quien desarrolló un rudimentario dispositivo de cálculo mecánico en 1830. John Atanasoff fue el primero que aplicó el procesamiento electrónico a la computación digital en 1939. En 1946, se implementó con válvulas de vacío una computadora digital electrónica denominada ENIAC. Aunque ocupaba una habitación entera, ENIAC no tenía ni siquiera la potencia de cálculo de una calculadora de bolsillo actual.

Dígitos binarios

Cada uno de los dos dígitos del sistema ***binario***, 1 y 0, se denomina ***bit***, que es la contracción de las palabras *binary digit* (dígito binario). En los circuitos digitales se emplean dos niveles de tensión diferentes para representar los dos bits. Por lo general, el 1 se representa mediante el nivel de tensión más elevado, que se denomina nivel ALTO (HIGH) y 0 se representa mediante el nivel de tensión más bajo, que se denomina nivel BAJO (LOW). Este convenio recibe el nombre de **lógica positiva** y es el que se va a emplear a lo largo del libro.

ALTO (HIGH) = 1 y BAJO (LOW) = 0

Un sistema en el que un 1 se representa por un nivel BAJO y un 0 mediante un nivel ALTO se dice que emplea *lógica negativa.*

Los grupos de bits (combinaciones de 1s y 0s), llamados *códigos*, se utilizan para representar números, letras, símbolos, instrucciones y cualquier otra cosa que se requiera en una determinada aplicación.

Niveles lógicos

Las tensiones empleadas para representar un 1 y un 0 se denominan *niveles lógicos*. En el caso ideal, un nivel de tensión representa un nivel ALTO y otro nivel de tensión representa un nivel BAJO. Sin embargo, en un circuito digital real, un nivel ALTO puede ser cualquier tensión entre un valor mínimo y un valor máximo especificados. Del mismo modo, un nivel BAJO puede ser cualquier tensión comprendida entre un mínimo y máximo especificados. No puede existir solapamiento entre el rango aceptado de niveles ALTO y el rango aceptado de niveles BAJO.

La Figura 1.5 ilustra el rango general de los niveles BAJO y ALTO aceptables para un circuito digital. La variable $V_{H(máx)}$ representa el valor máximo de tensión para el nivel ALTO y $V_{H(mín)}$ representa el valor de tensión mínimo para el nivel ALTO. El valor máximo de tensión para el nivel BAJO se representa mediante $V_{L(máx)}$ y el valor mínimo de tensión para el nivel BAJO mediante $V_{L(mín)}$. Los valores de tensión comprendidos entre $V_{L(máx)}$ y $V_{H(mín)}$ no son aceptables para un funcionamiento correcto. Una tensión en el rango no permitido puede ser interpretada por un determinado circuito tanto como un nivel ALTO cuanto como un nivel BAJO, por lo que no puede tomarse como un valor aceptable. Por ejemplo, los valores para el nivel ALTO en un determinado tipo de circuito digital denominado CMOS pueden variar en el rango de 2 V a 3,3 V y los valores para el nivel BAJO en el rango de 0 V a 0,8 V. De esta manera, si por ejemplo se aplica una tensión de 2,5 V, el circuito lo aceptará como un nivel ALTO, es decir, un 1 binario. Si se aplica una tensión de 0,5 V, el circuito lo aceptará como un nivel BAJO, es decir, un 0 binario. En este tipo de circuito, las tensiones comprendidas entre 0,8 V y 2 V no son aceptables.

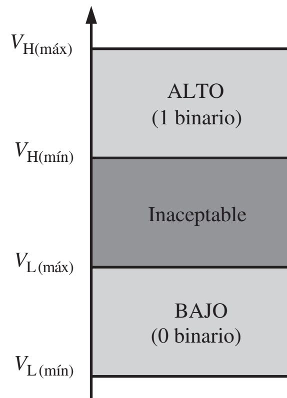


FIGURA 1.5 Rango de niveles lógicos de tensión para un circuito digital.

Formas de onda digitales

Las formas de onda digitales consisten en niveles de tensión que varían entre los estados o niveles ALTO y BAJO. La Figura 1.6(a) muestra que un **impulso** positivo se genera cuando la tensión (o la intensidad) pasa de su nivel normalmente BAJO hasta su nivel ALTO y luego vuelve otra vez a su nivel BAJO. El impulso negativo de la Figura 1.6(b) se genera cuando la tensión pasa de su nivel normalmente ALTO a su nivel BAJO y vuelve a su nivel ALTO. Una señal digital está formada por una serie de impulsos.

El impulso. Como se muestra en la Figura 1.6, un impulso tiene dos flancos: un **flanco anterior** que se produce en el instante t_0 y un **flanco posterior** que se produce en el instante posterior t_1. Para un impulso positivo, el flanco anterior es un flanco de subida y el flanco posterior es de bajada. Los impulsos mostrados en la Figura 1.6 son ideales porque se supone que los flancos de subida y de bajada ocurren en un tiempo cero (instantáneamente). En la práctica, estas transiciones no suceden de forma instantánea, aunque para la mayoría de las situaciones digitales podemos suponer que son impulsos ideales.

La Figura 1.7 muestra un impulso real (no ideal). En la práctica, todos los impulsos presentan alguna o todas de las características siguientes. En ocasiones, se producen picos de tensión y rizado debidos a los efectos capacitivos e inductivos parásitos. La caída puede ser provocada por las capacidades parásitas y la resistencia del circuito que forman un circuito *RC* con una constante de tiempo baja.

El tiempo requerido para que un impulso pase desde su nivel BAJO hasta su nivel ALTO se denomina **tiempo de subida** (t_r), y el tiempo requerido para la transición del nivel ALTO al nivel BAJO se denomina **tiempo de bajada** (t_f). En la práctica, el tiempo de subida se mide como el tiempo que tarda en pasar del 10% (altura respecto de la línea) al 90% de la amplitud del impulso y el tiempo de bajada se mide como el tiempo que tarda en pasar del 90% al 10% de la amplitud del impulso, como se puede ver en la Figura 1.7. La

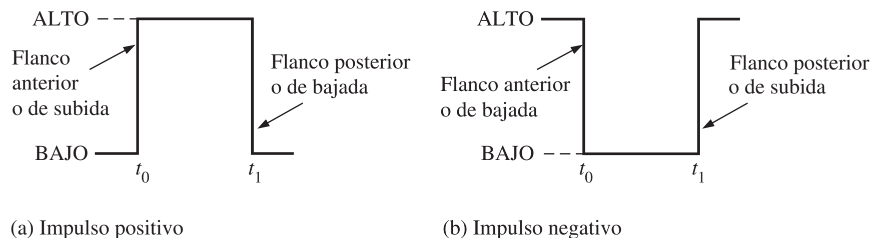


FIGURA 1.6 Impulsos ideales.

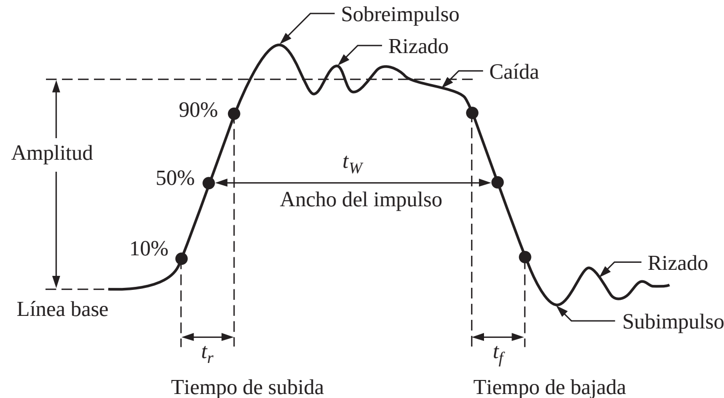


FIGURA 1.7 Características de los impulsos no ideales.

razón de que el 10% inferior y el 10% superior no se incluyan en los tiempos de subida y de bajada se debe a la no linealidad de la señal en esas áreas. El **ancho del impulso** (t_W) es una medida de la duración del impulso y, a menudo, se define como el intervalo de tiempo que transcurre entre los puntos en que la amplitud es del 50% en los flancos de subida y de bajada, como se indica en la Figura 1.7.

Características de la forma de onda. La mayoría de las formas de onda que se pueden encontrar en los sistemas digitales están formadas por series de impulsos, algunas veces denominados también *trenes de impulsos*, y pueden clasificarse en periódicas y no periódicas. Un tren de impulsos **periódico** es aquel que se repite a intervalos de tiempo fijos; este intervalo de tiempo fijo se denomina **período** (***T***). La **frecuencia** (***f***) es la velocidad a la que se repite y se mide en hercios (Hz). Por supuesto, un tren de impulsos no periódico no se repite a intervalos de tiempo fijos y puede estar formado por impulsos de distintos anchos y/o impulsos que tienen intervalos distintos de tiempo entre los pulsos. En la Figura 1.8 se muestra un ejemplo de cada tipo.

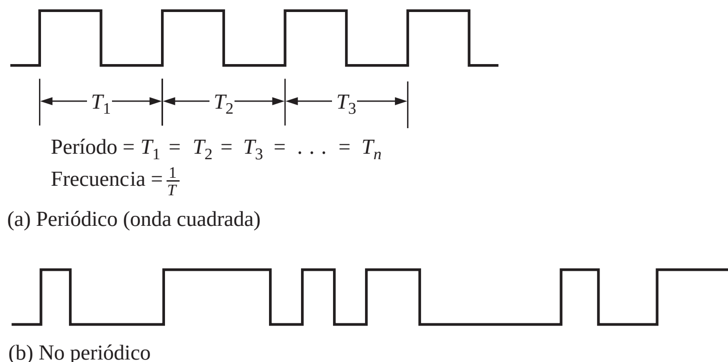


FIGURA 1.8 Ejemplos de formas de onda digitales.

La frecuencia (f) de un tren de pulsos (digital) es el inverso del período. La relación entre la frecuencia y el período se expresa como sigue:

Ecuación 1.1

$$f = \frac{1}{T}$$

Ecuación 1.2
$$T = \frac{1}{f}$$

Una característica importante de una señal digital periódica es su ***ciclo de trabajo***, que es el cociente entre el ancho del impulso (t_W) y el período (T) y puede expresarse como un porcentaje.

Ecuación 1.3
$$\text{Ciclo de trabajo} = \left(\frac{t_W}{T}\right)100\%$$

EJEMPLO 1.1

En la Figura 1.9 se muestra una parte de una señal digital periódica. Las medidas están expresadas en milisegundos. Determinar:
(a) período **(b)** frecuencia **(c)** ciclo de trabajo

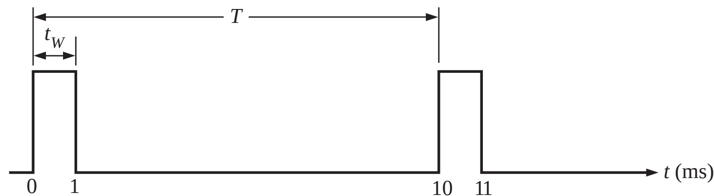


FIGURA 1.9

Solución

(a) El período se mide desde el flanco de uno de los impulsos hasta el correspondiente flanco del siguiente impulso. En este caso, T se mide desde el flanco de subida hasta el siguiente flanco de subida, como se indica en la figura, luego T es igual a **10 ms.**

(b) $$f = \frac{1}{T} = \frac{1}{10\,\text{ms}} = \mathbf{100\ Hz}$$

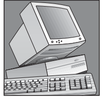
(c) $$\text{Ciclo de trabajo} = \left(\frac{t_W}{T}\right)100\% = \left(\frac{1\,\text{ms}}{10\,\text{ms}}\right)100\% = \mathbf{10}\%$$

Problema relacionado* Una señal digital periódica tiene un ancho de impulso de 25 ms y un período de 150 μs. Determinar la frecuencia y el ciclo de trabajo.

* Las respuestas se encuentran al final del capítulo.

Una señal digital contiene información binaria

La información binaria que manejan los sistemas digitales aparece en forma de señales que representan secuencias de bits. Cuando la señal está a nivel ALTO, quiere decir que está presente un 1 binario; cuando la señal está a nivel BAJO, lo indica un 0 binario. Cada bit dentro de una secuencia ocupa un intervalo de tiempo definido, denominado **período de bit**.



NOTAS INFORMÁTICAS

La velocidad a la que una computadora puede funcionar depende del tipo de microprocesador utilizado en el sistema. La especificación de velocidad, por ejemplo 3,5 GHz, de una computadora es la frecuencia máxima de reloj a la que el microprocesador puede trabajar.

El reloj. En los sistemas digitales, todas las señales están sincronizadas con una señal de temporización básica denominada ***reloj***. El reloj es una señal periódica en la que cada intervalo entre impulsos (el período) es igual a la duración de un bit.

En la Figura 1.10 se muestra un ejemplo de una señal de reloj. Observe que, en este caso, cada cambio de nivel de la señal *A* se produce en el flanco de subida de la señal de reloj. En otros casos, los cambios de nivel se producen en el flanco de bajada de dicha señal. Para cada duración de un bit de la señal de reloj, la forma de onda *A* se encuentra a nivel ALTO o bien a nivel BAJO. Como ya hemos mencionado, estos niveles ALTO y BAJO representan una secuencia de bits. Un grupo de varios bits se puede utilizar como parte de una información binaria, tal como un número o una letra. La señal de reloj en sí misma no transporta información.

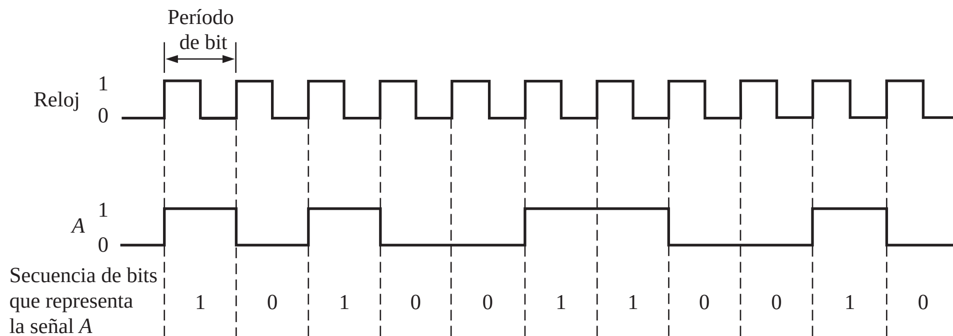


FIGURA 1.10 Ejemplo de una señal de reloj sincronizada con una señal que representa una secuencia de bits.

Diagramas de tiempos. Un ***diagrama de tiempos*** o **cronograma** es una gráfica de señales digitales que muestra la relación temporal real entre dos o más señales y cómo varía cada señal respecto a las demás. Al examinar un diagrama de tiempos, es posible determinar los estados (ALTO o BAJO) de todas las formas de onda en cualquier punto de tiempo especificado y el instante exacto en el que una forma de onda cambia de estado respecto a las restantes. La Figura 1.11 es un ejemplo de un diagrama de tiempos para cuatro señales.

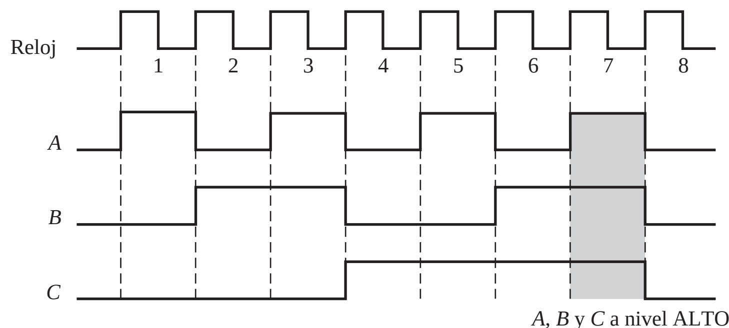


FIGURA 1.11 Ejemplo de un diagrama de tiempos.

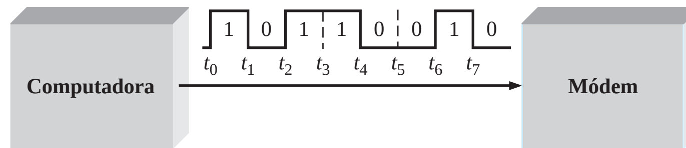
A partir de este diagrama de tiempos podemos ver, por ejemplo, que las tres formas de onda *A, B* y *C* están a nivel ALTO sólo durante el séptimo ciclo de reloj y las tres cambian de nuevo a nivel BAJO cuando termina dicho ciclo (área sombreada).

Transferencia de datos

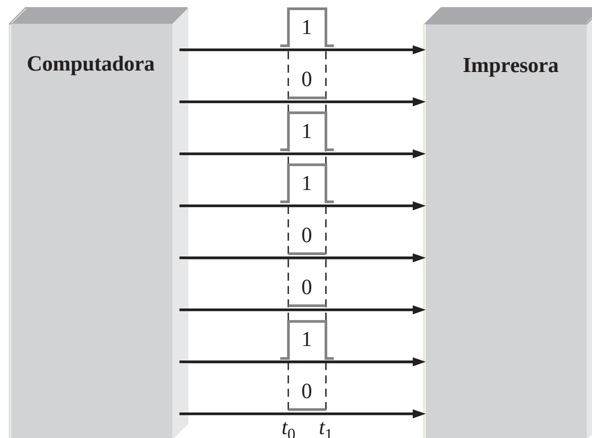
Los ***datos*** son grupos de bits que transportan algún tipo de información. Los datos binarios, que se representan mediante señales digitales, deben transferirse de un circuito a otro dentro de un sistema digital o desde un sistema a otro, para poder servir a un propósito determinado. Por ejemplo, los números almacenados en formato binario en la memoria de una computadora se deben transferir a la unidad central de procesamiento de la computadora para poder sumarse. El resultado de la suma debe entonces transferirse a la pantalla para visualizarse y/o enviarse de nuevo a la memoria. En los sistemas informáticos, como se muestra en la Figura 1.12, los datos binarios pueden transferirse de dos formas: en serie y en paralelo.

Cuando los bits se transmiten ***en serie*** de un punto a otro, se envían bit a bit a través de una sola línea, como se muestra en la Figura 1.12(a) para el caso de una transmisión computadora-módem. Durante el intervalo de tiempo de t_0 a t_1, se transmite el primer bit. Durante el intervalo de tiempo de t_1 a t_2, se transmite el segundo bit, y así sucesivamente. Por tanto, la transmisión de ocho de bits en serie precisa ocho intervalos de tiempo.

Cuando los bits se transmiten ***en paralelo***, todos los bits de un grupo se envían por líneas separadas al mismo tiempo. Como se muestra en la Figura 1.12(b) para el ejemplo de transmisión de ocho bits desde una computadora a una impresora, existe una línea para cada bit. Para transferir ocho bits en paralelo sólo se necesita un intervalo de tiempo frente a los ocho que se precisan en la transferencia en serie.



(a) Transferencia serie de 8 bits de datos binarios desde una computadora a un módem. El primer intervalo es de t_0 a t_1.



(b) Transferencia en paralelo de 8 bits de datos binarios desde una computadora a una impresora. t_0 es el instante inicial.

FIGURA 1.12 Transferencia en serie y en paralelo de datos binarios. Sólo se muestran las líneas de datos.

En resumen, la ventaja de una transmisión en serie de datos binarios es que sólo se necesita una línea. En la transmisión en paralelo se necesitan tantas líneas como número de bits que hay que transmitir al mismo tiempo. Uno de los inconvenientes de la transmisión en serie es que tarda más tiempo en transferir un número de bits dado que la transmisión en paralelo. Por ejemplo, si un bit puede transferirse en un 1 μs, entonces para transmitir 8 bits en serie se necesitan 8 μs, pero sólo 1 μs para hacerlo en paralelo. Una desventaja de la transmisión en paralelo es que se precisan más líneas.

EJEMPLO 1.2

(a) Determinar el tiempo total necesario para transferir en serie los ocho bits de la señal *A* mostrada en la Figura 1.13, e indicar la secuencia de bits. El bit más a la izquierda es el que se transmite en primer lugar. La señal de reloj de 10 kHz se emplea como referencia.

(b) ¿Cuál es el tiempo total para transmitir los mismos ocho bits en paralelo?

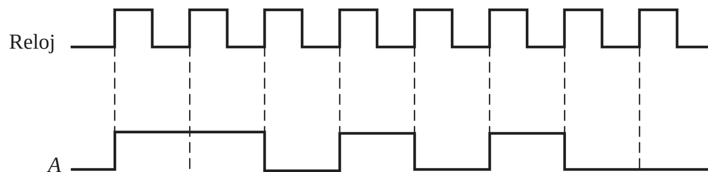


FIGURA 1.13

Solución

(a) Puesto que la frecuencia del reloj es de 100 kHz, el período es

$$T = \frac{1}{f} = \frac{1}{100\text{kHz}} = 10\,\mu\text{s}$$

Se necesitan 10 μs para transmitir cada bit de la señal. El tiempo total de transmisión para 8 bits es:

$$8 \times 10\,\mu\text{s} = \mathbf{80\,\boldsymbol{\mu}s}$$

Para determinar la secuencia de bits, examinamos la señal de la Figura 1.13 para cada período de bit. Si la señal *A* está a nivel ALTO durante el período de bit, se transmite un 1. Si la señal *A* está a nivel BAJO durante el período de bit, se transmite un 0. La secuencia de bits se muestra en la Figura 1.14. El bit más a la izquierda es el primero que se transmite.

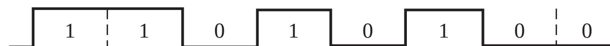


FIGURA 1.14

(b) Una transferencia en paralelo necesitaría **10 μs** para los ocho bits.

Problema relacionado Si los datos binarios se transmiten a una velocidad de 10 millones de bits por segundo (10 Mbits/s), ¿cuánto tiempo será necesario para transmitir en paralelo 16 bits por 16 líneas? ¿cuánto tiempo se tardaría en transmitir 16 bits en serie?

REVISIÓN DE LA SECCIÓN 1.2

1. Definir *binario.*
2. ¿Qué significa *bit*?
3. ¿Qué son los bits en un sistema binario?
4. ¿Cómo se miden el tiempo de subida y de bajada de un impulso?
5. Conociendo el período de una señal, ¿cómo se calcula la frecuencia?
6. Explicar qué es una señal de reloj.
7. ¿Cuál es el propósito de un cronograma o diagrama de tiempos?
8. ¿Cuál es la principal ventaja de la transmisión en paralelo de datos binarios frente a la transmisión en serie?

1.3 OPERACIONES LÓGICAS BÁSICAS

En su forma más simple, la lógica es la parte del razonamiento humano que nos dice que una determinada proposición (sentencia de asignación) es cierta si se cumplen ciertas condiciones. Las proposiciones se pueden clasificar como verdaderas o falsas. Muchas situaciones y procesos que encontramos en nuestra vida cotidiana pueden expresarse como funciones proposicionales o lógicas. Dado que tales funciones son sentencias verdaderas/falsas o afirmativas/negativas, pueden aplicarse a los circuitos digitales, ya que éstos se caracterizan por sus dos estados.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Enumerar las tres operaciones lógicas básicas. ■ Definir la operación NOT. ■ Definir la operación AND. ■ Definir la operación OR.

Cuando se combinan varias proposiciones se forman funciones lógicas o proposicionales. Por ejemplo, la proporción "la luz está encendida" será cierta si "la bombilla no está fundida" lo es y si "el interruptor está dado" también es verdadera. Por tanto, esta proposición lógica puede formularse de la manera siguiente: *la luz está encendida sólo si la bombilla no está fundida y el interruptor está dado.* En este ejemplo, la primera sentencia sólo es verdadera si las dos últimas lo son. La primera proposición ("la luz está encendida") es por tanto la proposición básica y las otras dos son las condiciones de las que depende la proposición.

Hacia 1850, el matemático y lógico irlandés George Boole desarrolló un sistema matemático para formular proposiciones lógicas con símbolos, de manera que los problemas puedan formularse y resolverse de forma similar a como se hace en el álgebra ordinaria. El álgebra de Boole, como se le conoce hoy día, encuentra aplicaciones en el diseño y el análisis de los sistemas digitales, y se tratará en detalle en el Capítulo 4.

El término ***lógico*** se aplica a los circuitos digitales que se utilizan para implementar funciones lógicas. Existen varios tipos de **circuitos** lógicos que son los elementos básicos que constituyen los bloques sobre los que se construyen los sistemas digitales más complejos, como por ejemplo una computadora. Ahora vamos a abordar estos elementos y vamos a estudiar sus funciones de una forma muy general. En capítulos posteriores estudiaremos estos circuitos en detalle.

En la Figura 1.15 se muestran los símbolos estándar distintivos de las tres operaciones lógicas básicas (NOT, AND y OR). Existen otros símbolos estándar para estas operaciones lógicas que se verán en el Capítu-

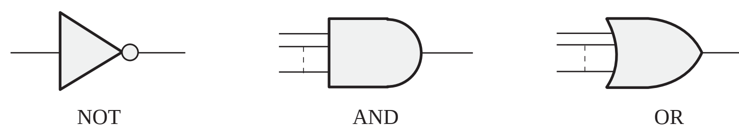


FIGURA 1.15 Operaciones lógicas básicas y sus símbolos.

lo 3. Las líneas conectadas a cada símbolo son las ***entradas*** y las ***salidas***. Las entradas son las líneas situadas a la izquierda de cada símbolo y la salida es la línea de la derecha. Un circuito que realiza una operación lógica determinada (NOT, AND, OR) se denomina ***puerta*** lógica. Las puertas AND y OR pueden tener cualquier número de entradas, como se indica mediante las líneas de puntos en la Figura 1.15.

En las operaciones lógicas, las condiciones mencionadas anteriormente de verdadero/falso se representan mediante un nivel ALTO (verdadero) y un nivel BAJO (falso). Cada una de las tres operaciones básicas da lugar a una única respuesta para un determinado conjunto de condiciones.

NOT

La operación ***NOT*** cambia de un nivel lógico al nivel lógico opuesto, como se muestra en la Figura 1.16. Cuando la entrada está a nivel ALTO (1), la salida se pone a nivel BAJO (0). Cuando la entrada está a nivel BAJO, la salida se pone a nivel ALTO. En cualquier caso, la salida *no* es la misma que la entrada. La operación NOT se implementa mediante un circuito lógico conocido como ***inversor***.



FIGURA 1.16 La operación NOT.

AND

La operación ***AND*** genera un nivel ALTO sólo cuando todas las entradas están a nivel ALTO, como se muestra en la Figura 1.17 para el caso de dos entradas. Cuando una entrada está a nivel ALTO *y* la otra entrada está a nivel ALTO, la salida se pone a nivel ALTO. Cuando cualquiera de las entradas o todas ellas están a nivel BAJO, la salida se pone a nivel BAJO. La operación AND se implementa mediante un circuito lógico conocido como *puerta AND.*

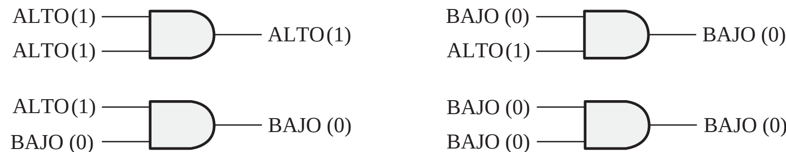


FIGURA 1.17 La operación AND.

OR

La operación ***OR*** genera un nivel ALTO cuando una o más entradas están a nivel ALTO, como se muestra en la Figura 1.18 para el caso de dos entradas. Cuando una de las entradas está a nivel ALTO *o* ambas entradas están a nivel ALTO, la salida es un nivel ALTO. Cuando ambas entradas están a nivel BAJO, la salida será un nivel BAJO. La operación OR se implementa mediante un circuito lógico denominado *puerta OR.*

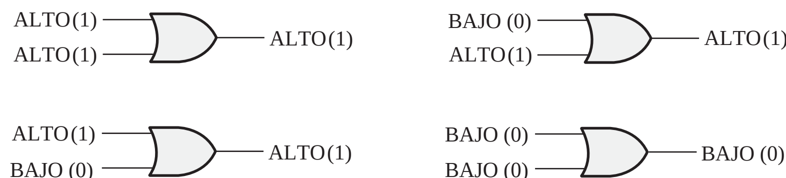


FIGURA 1.18 La operación OR.

REVISIÓN DE LA SECCIÓN 1.3

1. ¿Cuándo la operación NOT produce una salida a nivel ALTO?
2. ¿Cuándo la operación AND produce una salida a nivel ALTO?
3. ¿Cuándo la operación OR produce una salida a nivel ALTO?
4. ¿Qué es un inversor?
5. ¿Qué es una puerta lógica?

1.4 INTRODUCCIÓN A LAS FUNCIONES LÓGICAS BÁSICAS

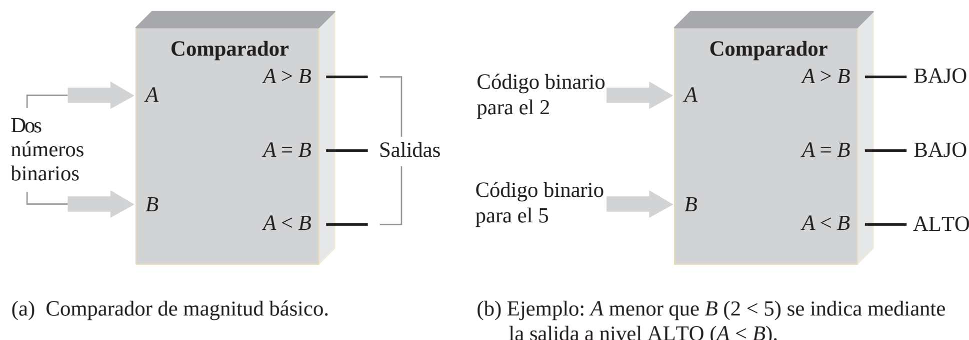
Los tres elementos lógicos básicos AND, OR y NOT se pueden combinar para formar circuitos lógicos más complejos, que realicen muchas operaciones útiles y que se empleen en la construcción de sistemas digitales completos. Algunas de las funciones lógicas más comunes son: comparación, aritmética, conversión de códigos, codificación, decodificación, selección de datos, almacenamiento y recuento. Esta sección proporciona una panorámica general de estas importantes funciones, con el fin de mostrarle cómo forman los bloques básicos de sistemas digitales, como por ejemplo las computadoras. Cada una de estas funciones básicas se verá en detalle en los capítulos siguientes.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Identificar ocho tipos básicos de funciones lógicas. ■ Describir un comparador de magnitud básico. ■ Enumerar las cuatro funciones aritméticas. ■ Describir un sumador básico. ■ Describir un comparador básico. ■ Describir un decodificador básico. ■ Definir multiplexación y demultiplexación. ■ Explicar cómo se lleva a cabo el almacenamiento de datos. ■ Describir la función de un contador básico.

La función de comparación

La **comparación de magnitudes** se realiza mediante un circuito lógico denominado **comparador**, que se estudia en el Capítulo 6. Su propósito es comparar dos cantidades e indicar si son iguales o no. Por ejemplo, supongamos que tenemos dos números y deseamos saber si son iguales o no; en el caso de que no sean iguales, queremos saber cuál es el mayor. La función de comparación se representa en la Figura 1.19. Se aplica un numero en formato binario (representado mediante niveles lógicos) a la entrada A y otro número binario (representado también mediante niveles lógicos) a la entrada B. Las salidas indican la relación entre los dos números, generando un nivel ALTO en la salida apropiada. Supongamos que una representación binaria del número 2 se aplica a la entrada A y que una representación binaria del número 5 se aplica a la entrada B



(a) Comparador de magnitud básico.

(b) Ejemplo: A menor que B ($2 < 5$) se indica mediante la salida a nivel ALTO ($A < B$).

FIGURA 1.19 La función de comparación.

(en el Capítulo 2 veremos la representación binaria de números y símbolos). Aparecerá un nivel ALTO en la salida $A < B$ (A menor que B), indicando la relación entre los dos números (2 menor que 5). Las flechas más anchas representan un grupo de líneas en paralelo a través de las que se transmiten los bits.

Funciones aritméticas

Suma. La adición se realiza mediante un circuito lógico llamada **sumador**, que es estudia en el Capítulo 6. Su función es sumar dos números binarios (que se aplican a las entradas A y B, junto con una entrada de acarreo C_{in}) y genera la suma (Σ) y un acarreo de salida (C_{out}), como se muestra en la Figura 1.20(a). La Figura 1.20(b) ilustra la suma de los números 3 y 9. Sabemos que la suma es 12; el sumador proporciona este resultado generando 2 en la salida suma y 1 en la salida de acarreo. En este ejemplo suponemos que la entrada de acarreo está a 0.

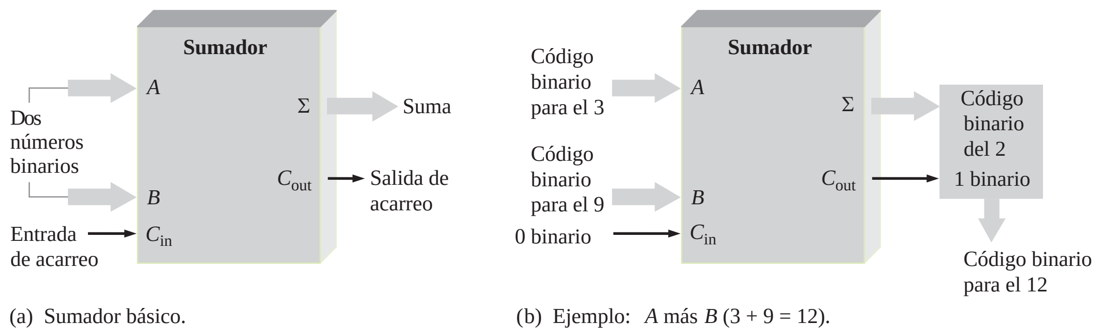
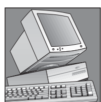


FIGURA 1.20 La función suma.

Resta. La sustracción se realiza también mediante un circuito lógico. Un **restador** requiere tres entradas: los dos números que se van a restar y una entrada de acarreo negativo (*borrow*). Las dos salidas corresponden a la diferencia y a la salida de acarreo negativo. Por ejemplo, cuando se resta 5 de 8 sin entrada de acarreo, la diferencia es 3 sin salida de acarreo. Como veremos en el Capítulo 2, la resta puede realizarse utilizando un sumador, ya que la operación de sustracción es simplemente un caso especial de la suma.

Multiplicación. La multiplicación se lleva a cabo con un circuito lógico llamado *multiplicador*. Dado que los números siempre se multiplican de dos en dos, serán necesarias dos entradas. La salida del multiplicador es el producto. Puesto que la multiplicación es simplemente una serie de sumas con desplazamientos de las posiciones de los productos parciales, se puede realizar utilizando un sumador junto con otros circuitos.

División. La división puede realizarse mediante una serie de sustracciones, comparaciones y desplazamientos, por lo que también puede efectuarse usando un sumador junto con otros circuitos. El divisor precisa dos entradas y las salidas generadas corresponden al cociente y al resto.



NOTAS INFORMÁTICAS

En un microprocesador, la unidad aritmético-lógica (UAL o ALU, *Arithmetic Logic Unit*) realiza las operaciones de suma, resta, multiplicación y división, así como las operaciones lógicas sobre los datos digitales mediante una serie de instrucciones. Una ALU típica se construye con muchos miles de puertas lógicas.

Función de conversión de código

Un **código** es un conjunto de bits ordenados de acuerdo a un modelo único y se emplea para representar información específica. Ejemplos de estas conversiones son las conversiones de binario a otros códigos, como por

ejemplo el código decimal binario (BCD, *Binary Coded Decimal*) y el código Gray. Algunos de estos códigos se estudian en el Capítulo 2 y en el Capítulo 6 se abordan los convertidores de código.

Función de codificación

La función de codificación se realiza mediante un circuito lógico denominado **codificador**, que se verá en el Capítulo 6. Un codificador convierte la información, como por ejemplo un número decimal o un carácter alfabético, en algún tipo de código. Por ejemplo, un cierto tipo de codificador convierte los dígitos decimales, de 0 a 9, a código binario. Un nivel ALTO en la entrada correspondiente a un determinado dígito decimal genera el código binario apropiado en las líneas de salida.

La Figura 1.21 es una sencilla ilustración de un codificador utilizado para convertir (codificar) una pulsación de una tecla de una calculadora en un código binario que puede ser procesado por los circuitos de la calculadora.

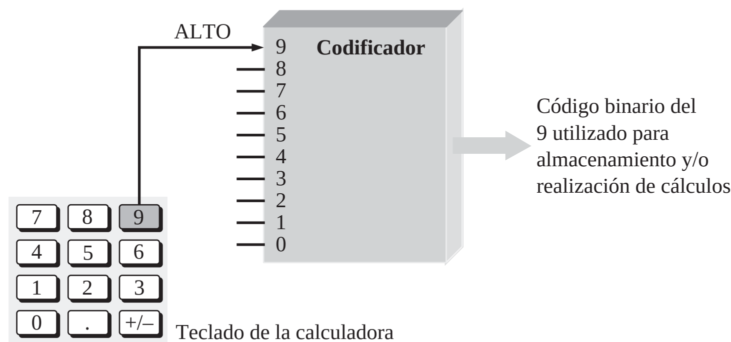


FIGURA 1.21 Un codificador utilizado para codificar una pulsación de una calculadora en un código binario que se almacenará o se empleará en los cálculos.

Función de decodificación

La función de decodificación se realiza mediante un circuito lógico llamado **decodificador**, que se verá en el Capítulo 6. Un decodificador convierte la información codificada, como puede ser un número binario, en otra información no codificada, como por ejemplo un número decimal. Por ejemplo, un determinado tipo de decodificador convierte un código binario de 4 bits en el correspondiente dígito decimal.

La Figura 1.22 es una sencilla ilustración de un tipo de decodificador que se emplea para activar un display de 7-segmentos. Cada uno de los siete segmentos del display está conectado a una línea de salida del decodificador. Cuando aparece un determinado código binario en las entradas del decodificador, se activan las correspondientes líneas de salida y se iluminan los segmentos apropiados del display para mostrar el dígito decimal que corresponde al código binario.

Función de selección de datos

Existen dos tipos de circuitos dedicados a la selección de datos: el multiplexor y el demultiplexor. El **multiplexor** es un circuito lógico que pasa los datos digitales procedentes de varias líneas de entrada a una única línea de salida según una secuencia de tiempos específica. Funcionalmente, un multiplexor puede representarse mediante una operación de conmutación electrónica que conecta secuencialmente cada una de las líneas de entrada a la línea de salida. El **demultiplexor** es un circuito que pasa los datos digitales procedentes de una línea de entrada a varias líneas de salida según una determinada secuencia de tiempo. En esencia, el demultiplexor es un multiplexor invertido.

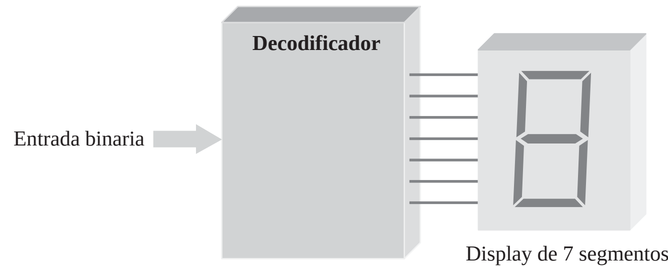


FIGURA 1.22 Un decodificador que convierte un código binario determinado en un código 7-segmentos que permite visualizar el número decimal correspondiente.

La multiplexación y la demultiplexación se utilizan cuando datos procedentes de distintas fuentes tienen que transmitirse a través de una línea hasta una localización distante y deben redistribuirse a varios destinos. La Figura 1.23 ilustra este tipo de aplicación en la que datos digitales procedentes de tres orígenes se envían a través de una sola línea hasta los tres terminales disponibles en la otra localización.

En la Figura 1.23, los datos de la entrada A se conectan a la línea de salida durante el intervalo de tiempo Δt_1 y se transmiten al demultiplexor que los pasa a la salida D. Luego, durante el intervalo de tiempo Δt_2, el multiplexor conmuta a la entrada B y el demultiplexor conmuta a la salida E. Durante el intervalo Δt_3, el multiplexor conmuta a la entrada C y el demultiplexor a la salida F.

En resumen, durante el primer intervalo de tiempo, los datos de la entrada A pasan a la salida D. Durante el segundo intervalo de tiempo, los datos de la entrada B pasan a la salida E y durante el tercer intervalo de tiempo, los datos de la entrada C pasan a la salida F. Después de esto, la secuencia se repite. Puesto que el tiempo se reparte entre varios orígenes y destinos, donde cada uno dispone de su turno para enviar y recibir datos, este proceso se denomina *multiplexación por división en el tiempo*.

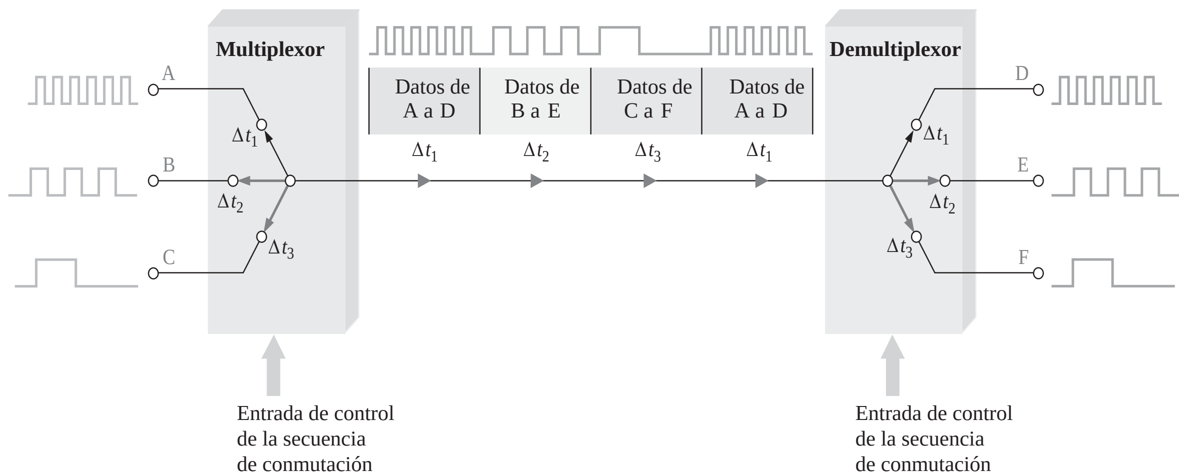


FIGURA 1.23 Ilustración de una aplicación básica de multiplexación/demultiplexación.

Función de almacenamiento

El **almacenamiento** es una función necesaria en la mayoría de los sistemas digitales y su propósito es el de conservar los datos binarios durante un período de tiempo. Algunos dispositivos de almacenamiento se utili-

zan para almacenamiento a corto plazo (temporal) y otros para almacenamiento a largo plazo (permanente). Un dispositivo de almacenamiento puede "memorizar" un bit o un grupo de bits y conservar la información tanto tiempo como sea necesario. Los tipos más comunes de dispositivos de almacenamiento son los flip-flops, los registros, las memorias semiconductoras, los discos magnéticos, las cintas magnéticas y los discos ópticos (CD).

Flip-flops. Un **flip-flop** es un circuito lógico biestable (dos estados estables) que sólo puede almacenar un bit cada vez, bien un 1 o un 0. La salida de un flip-flop indica qué bit está almacenado. Una salida a nivel ALTO indica que se ha almacenado un 1 y una salida a nivel BAJO indica que se ha almacenado un 0. Los flip-flops se implementan con puertas lógicas y se abordarán en detalle en el Capítulo 7.

Registros. Un **registro** se forma combinando varios flip-flops de manera que se puedan almacenar grupos de bits. Por ejemplo, un registro de 8 bits se construye a partir de ocho flip-flops. Además de para almacenar los bits, los registros pueden emplearse para desplazarlos de una posición a otra dentro del registro o fuera del mismo a otro circuito; por tanto, estos dispositivos se conocen como *registros de desplazamiento*, los cuales se estudian en el Capítulo 9.

Los dos tipos básicos de registros de desplazamiento son serie y paralelo. Los bits se almacenan en un registro de desplazamiento serie uno a uno, como se muestra en la Figura 1.24. Una buena analogía serían los pasajeros que entran en un autobús formando una única fila ante la puerta y salen del mismo modo.

En un registro paralelo los bits se almacenan simultáneamente a partir de líneas paralelo, como se muestra en la Figura 1.25. En este caso, una buena analogía serían los pasajeros que se montan en una montaña rusa, subiendo en los coches en paralelo.

Memorias semiconductoras. Las memorias semiconductoras son dispositivos típicamente utilizados para almacenar grandes cantidades de bits. En un tipo de memoria, denominado *memoria de sólo lectura* o ROM (*Read-Only Memory*), los datos se almacenan de forma permanente o semipermanente y no se pueden cambiar instantáneamente. En las memorias de acceso aleatorio o RAM (*Random Access Memory*), los datos binarios se almacenan temporalmente y puede cambiarse fácilmente. Las memorias se estudian en el Capítulo 10.

Memorias magnéticas. Las memorias de disco magnético se usan para el almacenamiento masivo de datos binarios. Ejemplos de estos dispositivos serían los disquetes utilizados en las computadoras y los discos duros

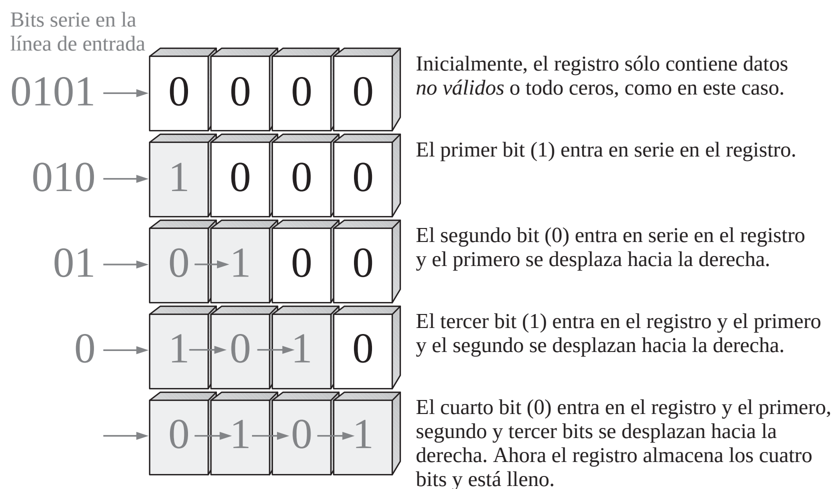


FIGURA 1.24 Ejemplo de funcionamiento de un registro de desplazamiento serie de 4 bits. Cada bloque representa una "celda" de almacenamiento o flip-flop.

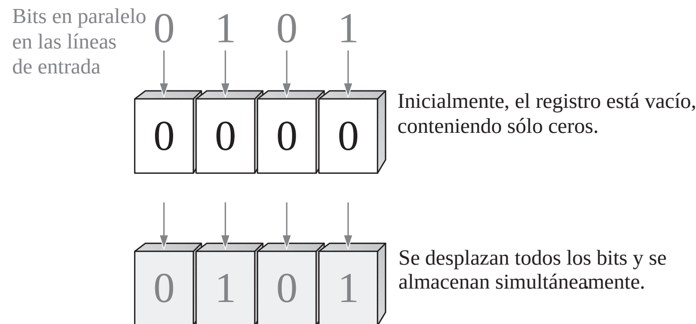
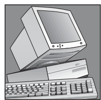


FIGURA 1.25 Ejemplo de funcionamiento de un registro de desplazamiento paralelo de 4 bits.

internos de las mismas. Los discos magneto-ópticos utilizan haces láser para almacenar y recuperar los datos. Las cintas magnéticas todavía se emplean en aplicaciones de memorias y para la realización de copias de seguridad de datos procedentes de otros dispositivos de almacenamiento.



NOTAS INFORMÁTICAS

Las memorias internas de las computadoras, RAM y ROM, así como las pequeñas memorias caché son memorias semiconductoras. Los registros de un microprocesador se construyen con flip-flops semiconductores. En las unidades internas de disco duro, las unidades de disquetes y de CD-ROM se emplean memorias de disco magnéticas.

Función de recuento

La función de recuento es importante en los sistemas digitales. Existen muchos tipos de **contadores** digitales, pero su objetivo básico es el de contar sucesos representados por cambios de nivel o por impulsos. Para realizar su función, el contador debe "recordar" el número actual, con el fin de poder pasar correctamente al siguiente número de la secuencia. Por tanto, la capacidad de almacenamiento es una característica importante de todos los contadores y, generalmente, se emplean los flip-flops para su implementación. La Figura 1.26 ilustra la idea básica del funcionamiento de un contador. Los contadores se estudian en el Capítulo 8.

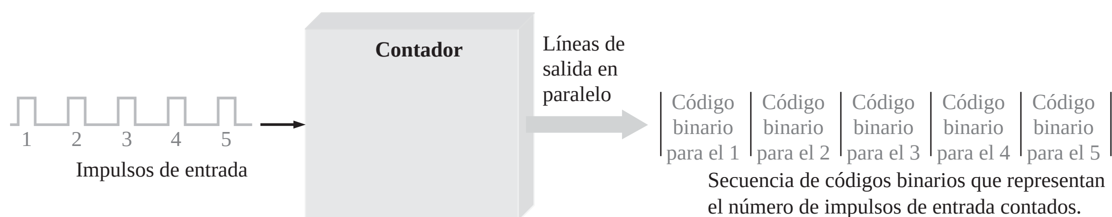


FIGURA 1.26 Ilustración del funcionamiento de un contador básico.

REVISIÓN DE LA SECCIÓN 1.4

1. ¿Qué hace un comparador?
2. ¿Cuáles son las cuatro funciones aritméticas básicas?
3. Describir qué es la codificación y proporcionar un ejemplo.
4. Describir qué es la decodificación y proporcionar un ejemplo.
5. Explicar el propósito básico de la multiplexación y la demultiplexación.

REVISIÓN DE LA SECCIÓN 1.4

6. Nombrar cuatro tipos de dispositivos de almacenamiento.

7. ¿Qué hace un contador?

1.5 CIRCUITOS INTEGRADOS DE FUNCIÓN FIJA

Todos los elementos y funciones lógicos que hemos tratado están disponibles como circuitos integrados (CI). Los sistemas digitales han incorporado circuitos integrados a lo largo de los años debido a su reducido tamaño, su alta fiabilidad, su bajo coste y su bajo consumo de potencia. Es importante ser capaz de reconocer los encapsulados de los CI y saber cómo se numeran sus pines, así como estar familiarizado con la forma en que la complejidad de los circuitos y su tecnología determinan las distintas clasificaciones de circuitos integrados.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Reconocer la diferencia entre los dispositivos de inserción y los dispositivos de montaje superficial. ■ Identificar los encapsulados DIP (*Dual In-line Package*). ■ Identificar los encapsulados SOIC (*Small-Outline Integrated Circuit*). ■ Identificar los encapsulados PLCC (*Plastic Leaded Chip Carrier*). ■ Identificar los encapsulados LCCC (*Leadless Ceramic Chip Carrier*). ■ Determinar la numeración de los pines en los distintos tipos de encapsulados de los CI. ■ Explicar la clasificación de los CI de función fija según su complejidad.

Un **circuito integrado (CI)** monolítico es un circuito electrónico construido enteramente sobre un pequeño chip de silicio. Todos los componentes que conforman el circuito: transistores, diodos, resistencias y condensadores, son parte integrante de un único chip. La lógica para funciones fijas y la lógica programable son las dos principales categorías en las que se enmarcan los CI digitales. En la lógica fija, las funciones lógicas son definidas por el fabricante y no es posible modificarlas.

La Figura 1.27 muestra una sección de un tipo de encapsulado de CI de función fija con el chip dentro del encapsulado. Los terminales del chip se conectan a los pines del encapsulado para permitir las conexiones de entrada y de salida al mundo exterior.

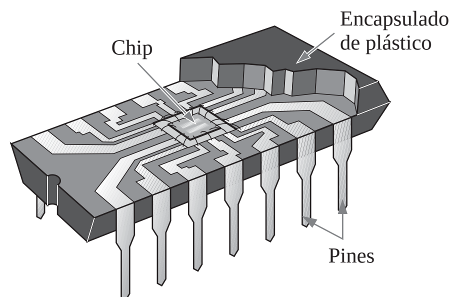


FIGURA 1.27 Sección de un encapsulado de CI de función fija que muestra el chip montado en el interior, con conexiones a los pines de entrada y de salida.

Encapsulados de CI

Los encapsulados de los CI se clasifican según la forma en que se montan sobre las tarjetas de circuito impreso (PCB, *Printed Circuit Board*) y pueden ser de inserción o de montaje superficial. Los encapsulados de

inserción disponen de pines (patas) que se introducen en los taladros de la tarjeta de circuito impreso y se sueldan a las pistas de la cara opuesta. El encapsulado de inserción más típico es el encapsulado **DIP** (*Dual Inline Package*), que se muestra en la Figura 1.28(a).

Otra técnica de encapsulado de CI es la tecnología de montaje superficial (**SMT**, *Surface-Mount Technology*). El montaje superficial representa una alternativa, que permite ahorrar espacio, al montaje de inserción. En la tecnología SMT, los taladros de las tarjetas de circuito impreso no son necesarios. Los pines de los encapsulados de montaje superficial se sueldan directamente a las pistas de una de las caras de la tarjeta, dejando la otra cara libre para añadir otros circuitos. Además, para un circuito con el mismo número de pines, un encapsulado de montaje superficial es mucho más pequeño que un encapsulado DIP, porque los pines se sitúan mucho más cercanos entre sí. Un ejemplo de encapsulado de montaje superficial es el circuito SOIC (*Small-Outline Integrated Circuit*) mostrado en la Figura 1.28(b).

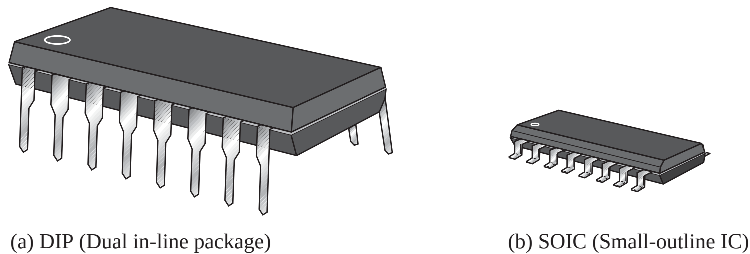


FIGURA 1.28 Ejemplos de dispositivos de inserción y de montaje superficial. El encapsulado DIP es más grande que el SOIC y tienen el mismo número de patas. Este DIP en concreto mide aproximadamente 2 cm de largo y el SOIC mide alrededor de 1 cm.

Los tres tipos de encapsulados SMT más comunes son el **SOIC** (small-outline IC), el **PLCC** (plastic leaded chip carrier) y el **LCCC** (leadless ceramic chip carrier). Estos tipos de encapsulados están disponibles en diferentes tamaños dependiendo del número de pines (cuanto más complejos son los circuitos, más pines son necesarios). En la Figura 1.29 se muestran ejemplos de cada uno de estos tipos. Como puede verse, los pines del SOIC tienen forma de “alas de gaviota”. Los pines del PLCC envuelven la parte inferior del encapsulado en forma de J. Por el contrario, los pines del LCCC tienen contactos metálicos que se introducen en el cuerpo cerámico. Otras variedades de los encapsulados SMT son el **SSOP** (*Shrink Small-Outline Package*), el **TSSOP** (*Thin Shrink Small-Outline Package*) y el **TVSOP** (*Thin Very Small-Outline Package*).

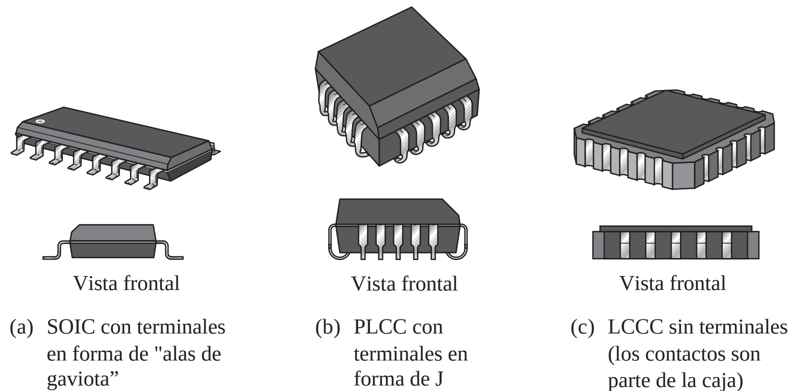


FIGURA 1.29 Ejemplos de configuraciones de encapsulados de montaje superficial.

Numeración de los pines

Todos los encapsulados de CI utilizan un formato estándar para numerar los pines (terminales). Para un encapsulado de 16 pines, los tipos DIP y SOIC tienen la disposición que se indica en la Figura 1.30(a). En la parte superior del encapsulado, se indica el pin1 mediante identificador que puede ser un pequeño punto, una muesca o un borde biselado. Además, con la muesca orientada hacia arriba, el pin 1 siempre es el pin situado más a la izquierda, como se indica. Comenzando por el pin 1, el número de pin aumenta a medida que se desciende y se continúa por el lado opuesto en sentido ascendente. El número mayor de pin es siempre el situado a la derecha de la muesca o el que está enfrente del punto.

Los encapsulados PLCC y LCCC tienen terminales en sus cuatro costados. El pin 1 se indica mediante un punto u otra marca y se sitúa en el centro de uno cualquiera de los lados del chip. La numeración de los terminales asciende en sentido contrario a las agujas del reloj mirando la parte superior del encapsulado. El pin de mayor numeración está siempre a la derecha del pin 1. La Figura 1.30(b) ilustra este formato para un encapsulado PLCC de 20 pines.

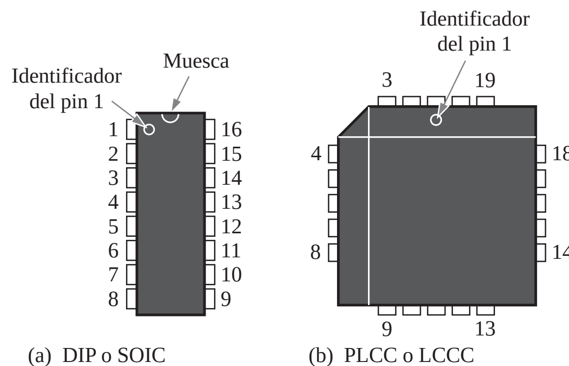


FIGURA 1.30 Numeración de los pines para dos tipos estándar de encapsulados de CI (vistas superiores).

Clasificación de los CI de función fija según su complejidad

Los circuitos integrados digitales de función fija se clasifican según su complejidad. A continuación se enumeran de menor a mayor complejidad. La clasificación por complejidad establecida aquí para SSI, MSI, LSI, VLSI y ULSI está generalmente aceptada, aunque las definiciones pueden variar de una fuente de información a otra.

- **Integración a pequeña escala** (SSI, *Small-Scale Integration*). Describe los CI de función fija que contienen hasta diez puertas equivalentes en un mismo chip, e incluyen puertas básicas y flip-flops.
- **Integración a media escala** (MSI, *Medium-Scale Integration*). Describe los CI que contienen entre 10 y 100 puertas equivalentes en un mismo chip. Incluyen funciones lógicas como codificadores, decodificadores, contadores, registros, multiplexores, circuitos aritméticos, memorias pequeñas y otras.
- **Integración a gran escala** (LSI, *Large-Scale Integration*). Es una categoría de los CI que incluyen entre 100 y 10.000 puertas equivalentes por chip, incluyendo memorias.
- **Integración de muy gran escala** (VLSI, *Very Large-Scale Integration*). Describe los CI con un número de puertas equivalentes desde 10.000 hasta 100.000 por chip.
- **Integración a ultra escala** (ULSI, *Ultra Large-Scale Integration*). Describe memorias de gran capacidad, grandes **microprocesadores** y computadoras en un solo chip. Esta categoría designa los CI que contienen más de 100.000 puertas equivalentes por chip.

Tecnologías de circuitos integrados

Los tipos de transistores con los que se implementan los circuitos integrados pueden ser transistores bipolares o MOSFET (*Metal-Oxide Semiconductor Field-Effect Transistor*, transistor de efecto de campo por unión metal-óxido-semiconductor). Una tecnología de circuitos que utiliza MOSFET es la tecnología CMOS (*Complementary* MOS, MOS complementario). Un tipo de tecnología de CI de función fija que utiliza los transistores bipolares es la TTL (*Transistor-Transistor Logic*, lógica transistor-transistor). BiCMOS utiliza una combinación de las tecnologías CMOS y TTL.

Todas las puertas y otras funciones se pueden implementar con cualquier tipo de tecnología de circuitos. Generalmente, los circuitos SSI y MSI están disponibles en CMOS y en TTL. LSI, VLSI y ULSI suelen implementarse con tecnología CMOS o NMOS, porque requieren una menor superficie de chip y consumen menos potencia. En el Capítulo 3 se tratan más detalladamente estas tecnologías de circuitos integrados. Además, el Capítulo 14 proporciona información completa a nivel de circuito.

Precauciones para la manipulación de dispositivos CMOS. Debido a su estructura, los dispositivos CMOS son muy sensibles a las cargas estáticas y pueden resultar dañados por las descargas electrostáticas si no se manipulan correctamente. Al trabajar con dispositivos CMOS deberán tomarse las siguientes precauciones:

- Los dispositivos CMOS deben ser suministrados y almacenados en espuma conductiva.
- Todos los instrumentos y bancos metálicos utilizado en las prueban deberán conectarse a una toma de tierra.
- Las herramientas de trabajo deben conectarse a tierra a través de un cable y resistencias en serie de alto valor.
- No debe retirarse un dispositivos CMOS (o cualquier dispositivo) de un circuito mientras que la alimentación continua esté conectada.
- No deben conectarse tensiones de señal o corriente a un dispositivo CMOS cuando la alimentación continua esté apagada.

REVISIÓN DE LA SECCIÓN 1.5

1. ¿Qué es un circuito integrado?
2. Defina los términos DIP, SMT, SOIC, SSI, MSI, LSI, VLSI y ULSI.
3. En general, ¿dentro de qué categoría se encuentra un CI de función fija con el siguiente número de puertas equivalentes?

   (a) 10 (b) 75 (c) 500 (d) 15.000 (e) 200.000

1.6 INTRODUCCIÓN A LA LÓGICA PROGRAMABLE

La lógica programable requiere tanto hardware como software. Los dispositivos lógicos programables pueden programarse para que el fabricante o el usuario pueda llevar a cabo funciones lógicas específicas. Una ventaja de la lógica programable frente a la lógica fija es que los dispositivos utilizan menos espacio de la tarjeta de circuito impreso para una cantidad equivalente de lógica. Otra ventaja es que, con la lógica programable, los diseños se pueden modificar fácilmente sin tener que recablear o reemplazar componentes. Además, generalmente un diseño lógico se puede implementar más rápidamente y con menos coste utilizando circuitos lógicos programables en lugar de los CI de función fija.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Establecer los principales tipos de dispositivos lógicos programables y comentar las diferencias. ■ Comentar los métodos de programación. ■ Enumerar los principales lenguajes de programación

utilizados en la lógica programable. ■ Definir el proceso de diseño de los dispositivos lógicos programables.

Tipos de dispositivos lógicos programables

Existen muchos tipos de dispositivos lógicos programables, desde pequeños dispositivos que pueden reemplazar a algunos de los dispositivos de función fija hasta complejos dispositivos de alta densidad que pueden reemplazar a miles de dispositivos de función fija. Las dos principales categorías de los dispositivos lógicos programables de usuario son los **PLD** (*Programmable Logic Device*, dispositivo lógico programable) y las **FPGA** (*Field Programmable Gate Array*, matrices de puertas programable por campo), que se muestran en la Figura 1.31. Los PLD pueden ser SPLD (*Simple* PLD, PLD simple) o CPLD (*Complex* PLD, PLD complejo).

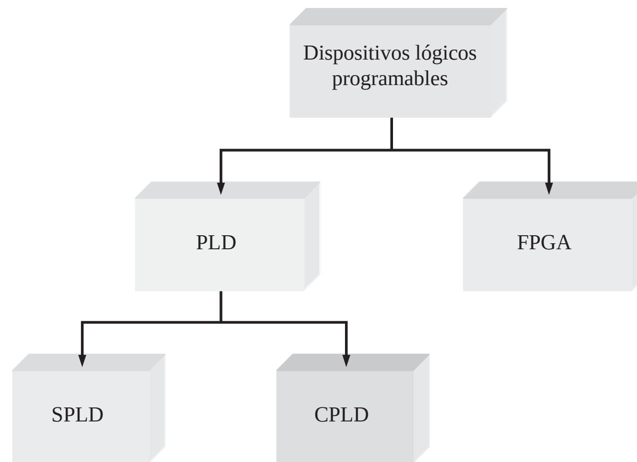


FIGURA 1.31 Lógica programable.

SPLD (Simple Programmable Logic Device). El SPLD se corresponde con el dispositivo lógico programable (PLD) original y todavía está disponible para aplicaciones de pequeña escala. Generalmente, un **SPLD** puede reemplazar a diez CI de función fija y sus interconexiones, dependiendo del tipo de funciones y del SPLD específico. La mayoría de los SPLD pertenecen a una de dos posibles categorías: PAL y GAL. Una **PAL** (*Programmable Array Logic*, matriz lógica programable) es un dispositivo que se puede programar una vez. Consta de una matriz programable de puertas AND y una matriz fija de puertas OR, como se muestra en la Figura 1.32(a). Una **GAL** (*Generic Array Logic*, matriz lógica genérica) es un dispositivo que es básicamente una PAL que puede reprogramarse muchas veces. Consta de una matriz reprogramable de puertas AND y de una matriz fija de puertas OR con salidas programables, como se muestra en la Figura 1.32(b). En la Figura 1.33 se muestra el encapsulado típico de un SPLD, que normalmente dispone de entre 24 y 28 pines.

CPLD (Complex Programmable Logic Device). A medida que la tecnología progresaba y que la cantidad de circuitería que se podía meter en un chip (densidad del chip) aumentaba, los fabricantes fueron capaces de incluir más de un SPLD en un mismo chip, lo que dio lugar al nacimiento del CPLD. En esencia, un **CPLD** es un dispositivo que contiene varios SPLD y que puede reemplazar a muchos CI de función fija. La Figura 1.34 muestra un diagrama de bloques básico de un CPLD con cuatro bloques de matriz lógica (LAB, *Logic Array Block*) y una PIA (*Programmable Interconnection Array*, matriz de interconexión programable). Dependiendo del CPLD específico, puede contener desde dos hasta sesenta y cuatro bloques LAB. Cada matriz lógica es aproximadamente equivalente a un SPLD.

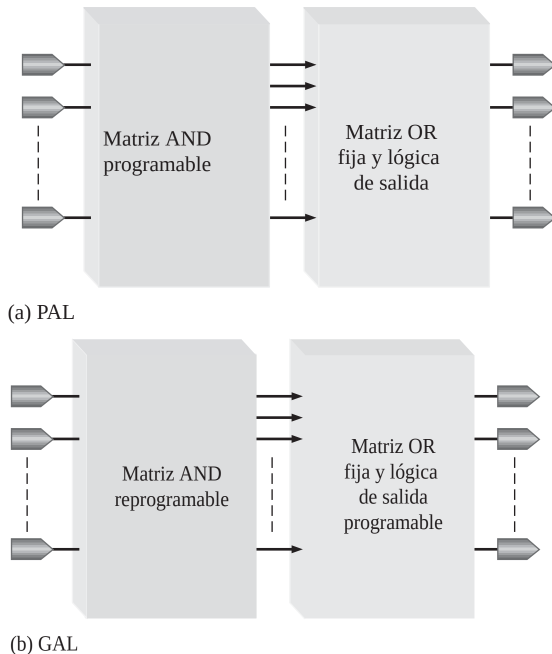


FIGURA 1.32 Diagramas de bloques de los dispositivos lógicos programables SPLD.

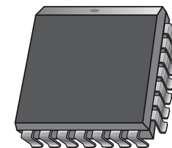


FIGURA 1.33 Encapsulado típico del SPLD.

Generalmente, los CPLD pueden utilizarse para implementar cualquiera de las funciones lógicas estudiadas anteriormente, como por ejemplo decodificadores, codificadores, multiplexores, demultiplexores y sumadores. Además, hay disponibles diversas configuraciones que normalmente emplean encapsulados con 44 hasta 160 pines. En la Figura 1.35 se muestran ejemplos de encapsulados de CPLD.

FPGA (Field Programmable Gate Array). Por lo general, una **FPGA** es más compleja y tiene una densidad mucho mayor que un CPLD, aunque en ocasiones sus aplicaciones pueden solaparse. Como hemos dicho, el SPLD y el CPLD están íntimamente relacionados, puesto que, básicamente, el CPLD contiene un conjunto de dispositivos SPLD. Sin embargo, la FPGA tiene una estructura interna (arquitectura) diferente, como se ilustra en la Figura 1.36. Los tres elementos básicos en una FPGA son el bloque lógico, las interconexiones programables y los bloques de entrada/salida (E/S). Los bloques lógicos de una FPGA no son tan complejos como los bloques de matrices lógicas (LAB) de un CPLD, aunque generalmente contienen muchos más. Cuando los bloques lógicos son relativamente simples, la arquitectura de la FPGA se dice que es de *granularidad fina*. Cuando los bloques lógicos son grandes y más complejos, la arquitectura se denomina de *granularidad gruesa.* Los bloques de E/S se encuentran en los bordes exteriores de la estructura y proporcionan entrada, salida o acceso bidireccional seleccionable al mundo exterior. La matriz de interconexiones programable distribuida proporciona la interconexión de los bloques lógicos y la conexión a las entradas y las salidas. Las FPGA grandes pueden contener decenas de miles de bloques lógicos además de memoria y otros recursos. En la Figura 1.37 se muestra un encapsulado BGA (*Ball-Grid Array*) típico para FPGA. Estos tipos de encapsulados pueden tener unos 1.000 pines de entrada y salida.

El proceso de programación

Puede pensarse en un SPLD, un CPLD o una FPGA como en un “circuito en blanco” en el que se va a implementar un circuito o sistema específico utilizando un determinado proceso. Este proceso requiere tener insta-

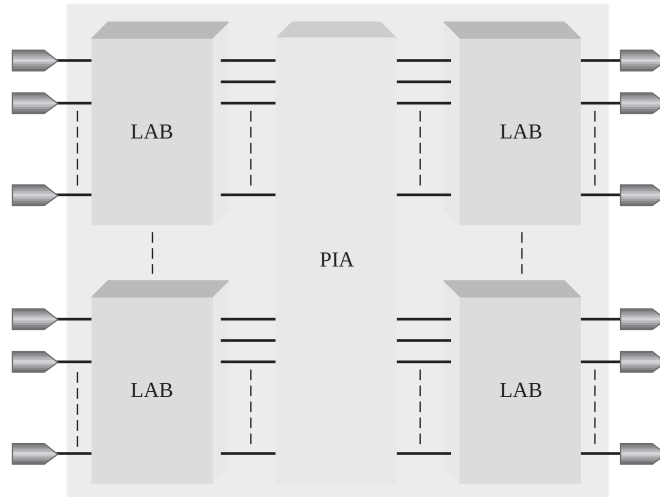


FIGURA 1.34 Diagrama de bloques general de un CPLD.

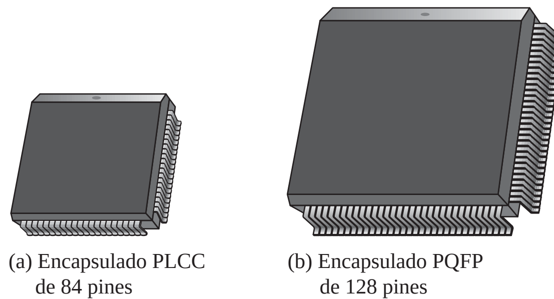


FIGURA 1.35 Encapsulados típicos de los dispositivos CPLD.

lado un paquete de desarrollo software en una computadora que permita implementar un diseño de circuito en el chip programable. Las computadoras deben poder interaccionar con una tarjeta de desarrollo o con una utilidad de programación que contenga el dispositivo, como se ilustra en la Figura 1.38.

En el proceso de implementar un diseño lógico digital en un dispositivo lógico programable son necesarios varios pasos, lo que se denomina *diagrama de flujo del diseño.* En la Figura 1.39 se presenta un diagrama de bloques de un proceso típico de programación. Como se indica, el flujo de diseño tiene acceso a una biblioteca de diseño.

Introducción del diseño. Este es el primer paso de programación. El diseño del circuito o sistema debe introducirse en el software de diseño utilizando un medio de entrada basado en texto, en gráficos (captura de esquemáticos) o en una descripción del diagrama de estados. El método para introducir el diseño es independiente del dispositivo. La introducción del diseño por técnicas basadas en texto se lleva a cabo mediante un lenguaje de descripción del hardware (HDL, *Hardware Description Language*), como por ejemplo VHDL, Verilog, AHDL o ABEL. La interfaz gráfica (esquemáticos) permite seleccionar funciones lógicas prealmacenadas en una biblioteca, colocarlas en la pantalla e interconectarlas para crear un diseño lógico. La introducción del diagrama de estados requiere especificar los estados por los que pasa el circuito lógico secuencial, así como las condiciones que dan lugar a cada cambio de estado.

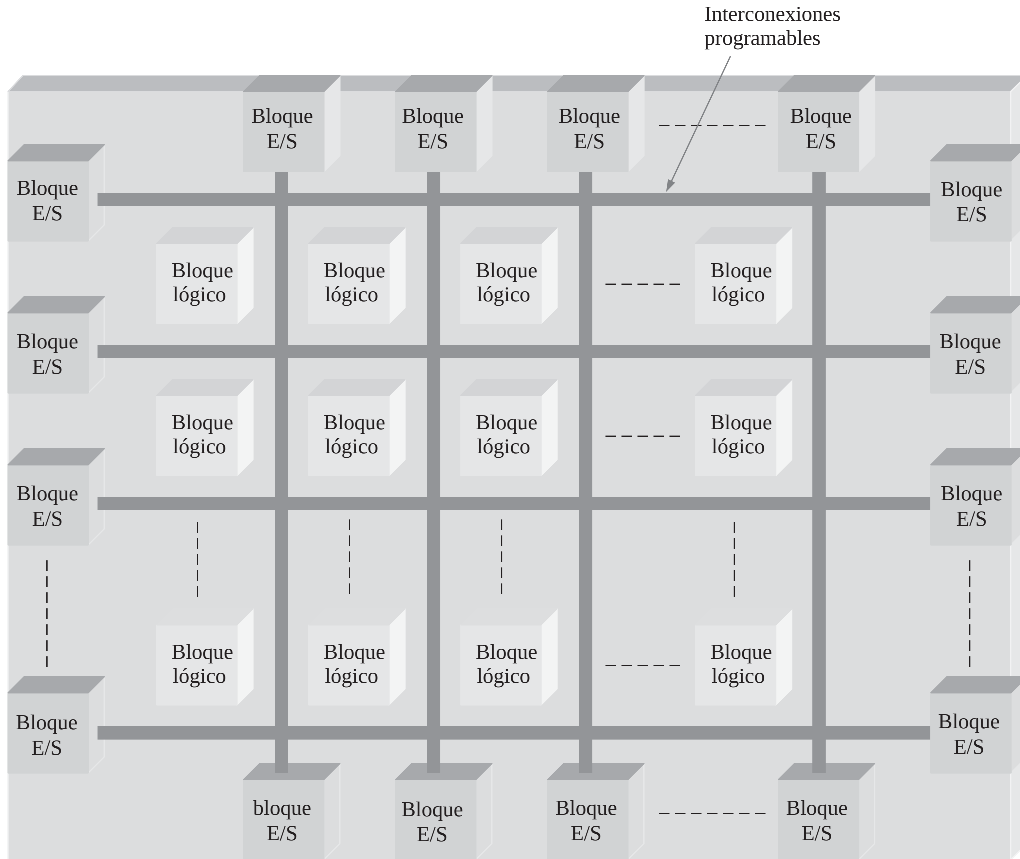


FIGURA 1.36 Estructura básica de una FPGA.

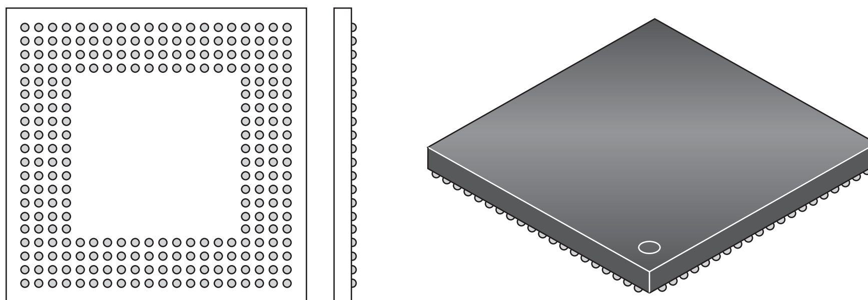


FIGURA 1.37 Una configuración típica de encapsulado BGA (ball-grid array).

Una vez que se ha introducido el diseño, se compila. Un **compilador** es un programa que controla el proceso del flujo de diseño y traduce el código fuente en código objeto en un formato que puede ser probado lógicamente o descargado en el dispositivo. El código fuente se crea durante la fase de introducción del diseño y el código objeto es el código final que realmente hace que el diseño pueda implementarse en el dispositivo programable.

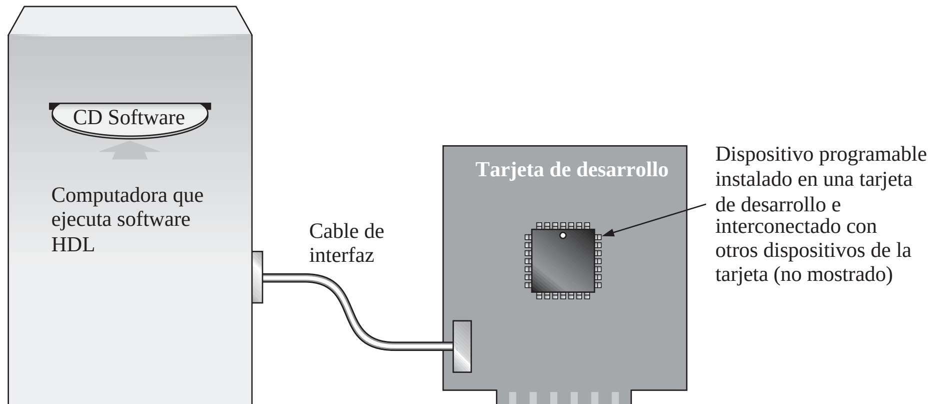


FIGURA 1.38 Configuración básica para la programación de un PLD o una FPGA.

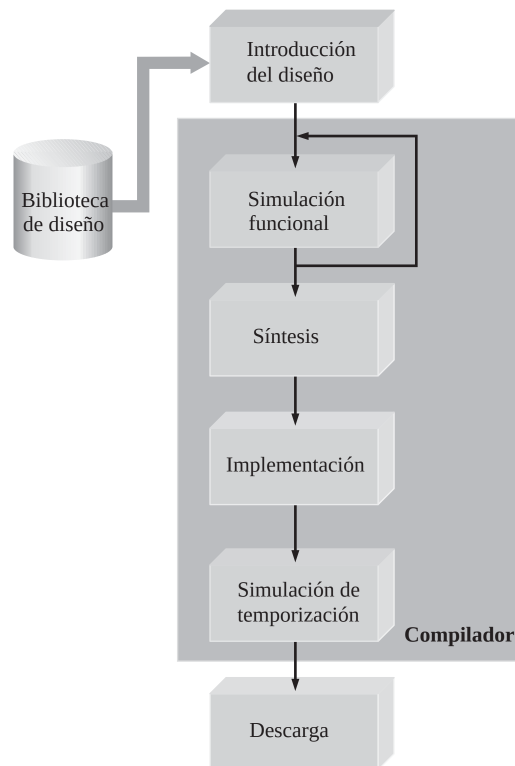


FIGURA 1.39 Organigrama básico del diseño del dispositivos lógicos programables.

Simulación funcional. El diseño introducido y compilado se simula por software para confirmar que el circuito lógico funciona como se esperaba. La simulación verificará que se generan las salidas correctas para un conjunto específico de entradas. Generalmente, una herramienta software independiente del dispositivo que lleva a cabo esta función se denomina *editor de señales.* Cualquier fallo que indique la simulación deberá corregirse realizando los cambios apropiados en la fase de introducción del diseño.

Síntesis. La **síntesis** es el proceso mediante el cual el diseño se traduce en una lista de componentes (*netlist*), la cual tiene un formato estándar y es independiente del dispositivo.

Implementación. La **implementación** es la fase en la que las estructuras lógicas descritas en la *netlist* se hacen corresponder con la estructura real del dispositivo específico que se va a programar. El proceso de implementación se denomina *colocación y rutado* y como resultado proporciona una salida denominada flujo de bits (*bitstream*), que es dependiente del dispositivo.

Simulación de temporización. Este paso se realiza después de que el diseño se haya mapeado sobre el dispositivo específico. Esta simulación se utiliza fundamentalmente para confirmar que no existen fallos de diseño o problemas de temporización debidos a los retardos de propagación.

Descarga. Una vez que se ha generado el flujo de bits para un dispositivo programable concreto, hay que descargarlo en el dispositivo para implementar el diseño software en el hardware. Algunos dispositivos programables tienen que instalarse sobre un equipo especial, denominado *programador de* dispositivos, o sobre una tarjeta de desarrollo. Otros dispositivos pueden programarse dentro de un sistema (lo que se denomina programación dentro del sistema, ISP, *In-System Programming*) utilizando una interfaz estándar JTAG (*Joint Test Action Group*). Algunos dispositivos son volátiles, lo que quiere decir que su contenido se pierde cuando se ponen a cero o se quita la alimentación. En este caso, los datos del flujo de bits deben almacenarse en una memoria y cargarse de nuevo en el dispositivo después de cada puesta a cero o desconexión de la alimentación. Además, el contenido de un dispositivo ISP se puede manipular o actualizar mientras que el sistema está funcionando. Esto se denomina reconfiguración “sobre la marcha”.

REVISIÓN DE LA SECCIÓN 1.6

1. Enumerar las tres categorías principales de dispositivos lógicos programables e indicar sus acrónimos.
2. ¿En qué se diferencia un CPLD de un SPLD?
3. Enumere los pasos del proceso de programación.
4. Explique brevemente cada uno de los pasos enumerados en la pregunta anterior.

1.7 INSTRUMENTOS DE MEDIDA Y PRUEBA

La **localización de averías** es el proceso de aislar, identificar y corregir de forma sistemática un fallo en un circuito o sistema. Existe una gran variedad de instrumentos que se pueden utilizar en la localización de averías y la realización de pruebas. En esta sección, se presentan y exponen algunos equipos típicos.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Diferenciar entre un osciloscopio analógico y uno digital. ■ Reconocer los controles más comunes del osciloscopio. ■ Determinar la amplitud, el período, la frecuencia y el ciclo de trabajo de una señal de impulsos con un osciloscopio. ■ Explicar el analizador lógico y algunos de sus formatos más comunes. ■ Describir el propósito de una fuente de alimentación de continua, de un generador de funciones y de un multímetro digital.

El osciloscopio

El osciloscopio es uno de los instrumentos más ampliamente utilizado para la realización de pruebas y la localización de averías. Básicamente, el osciloscopio es un dispositivo con pantalla gráfica que traza una gráfica de una señal eléctrica en su pantalla. En la mayor parte de las aplicaciones, las gráficas se muestran como

señales en función del tiempo. El eje vertical de la pantalla representa la tensión y el eje horizontal representa el tiempo. La amplitud, el período y la frecuencia de una señal se pueden medir con el osciloscopio. Además, pueden determinarse el ancho del impulso, el ciclo de trabajo, el tiempo de subida y el tiempo de bajada de una señal de impulsos. La mayoría de los osciloscopios pueden mostrar a la vez al menos dos señales en la pantalla, lo que permite observar su relación en el tiempo. En la Figura 1.40 se muestra un osciloscopio típico.



FIGURA 1.40 Osciloscopio típico de doble canal. Reproducida con permiso de Tektronix, Inc.

Para visualizar formas de onda digitales se pueden emplear dos tipos de osciloscopios: análogico y digital. Como se muestra en la Figura 1.41(a), el osciloscopio analógico funciona aplicando directamente la señal que se va a medir para controlar el movimiento de arriba a abajo del haz de electrones del tubo de rayos catódicos (TRC) a medida que oscila a lo largo de la pantalla. De este modo, el haz traza la forma de onda en la pantalla. Como se ilustra en la Figura 1.41(b), el osciloscopio digital convierte la forma de onda que se va a medir en información digital mediante un proceso de muestreo que se realiza en un convertir analógico-digital (ADC, *Analog-to-Digital Converter*). A continuación, la información digital se utiliza para reconstruir la forma de onda en la pantalla.

El osciloscopio digital se utiliza mucho más que el analógico. Sin embargo, en muchas aplicaciones puede utilizarse cualquiera de ellos, ya que cada uno tiene características que le hacen más adecuado para cada situación concreta. Un osciloscopio analógico muestra las formas de onda tal y como se producen en "tiempo real". Los osciloscopios digitales resultan útiles para medir impulsos transitorios que pueden producirse de forma aleatoria o sólo una vez. También, puesto que la información sobre la forma de onda medida se puede almacenar en un osciloscopio digital, puede visualizarse en cualquier instante posterior, imprimirse o analizarse en profundidad utilizando una computadora o cualquier otro medio.

Operación básica de los osciloscopios analógicos. Para medir una tensión, debe conectarse una **sonda** al punto del circuito en el que está presente la tensión. Generalmente, se utiliza una sonda ×10 que reduce (atenúa) la amplitud de la señal en un factor de diez. La señal atraviesa la sonda por sus circuitos verticales donde bien es atenuada o amplificada, dependiendo de la amplitud real y de dónde se haya colocado el control vertical del osciloscopio. Los circuitos verticales excitan entonces las placas de deflexión verticales del TRC. La señal pasa a los circuitos de disparo (*trigger*) que activan los circuitos horizontales para iniciar el barrido horizontal repetitivo del haz de electrones a lo largo de la pantalla usando una señal en forma de diente de sierra.

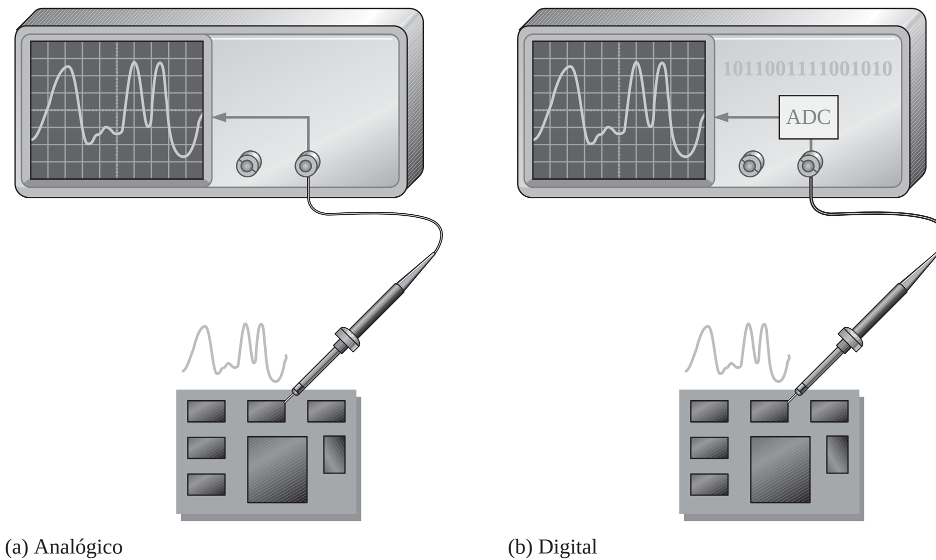


FIGURA 1.41 Comparación de los osciloscopios analógico y digital.

Hay disponibles muchos valores para el número de barridos por segundo, con el fin de que el haz parezca formar una línea sólida a lo largo de la pantalla sobre la forma de la señal. En la Figura 1.42 se muestra esta operación básica.

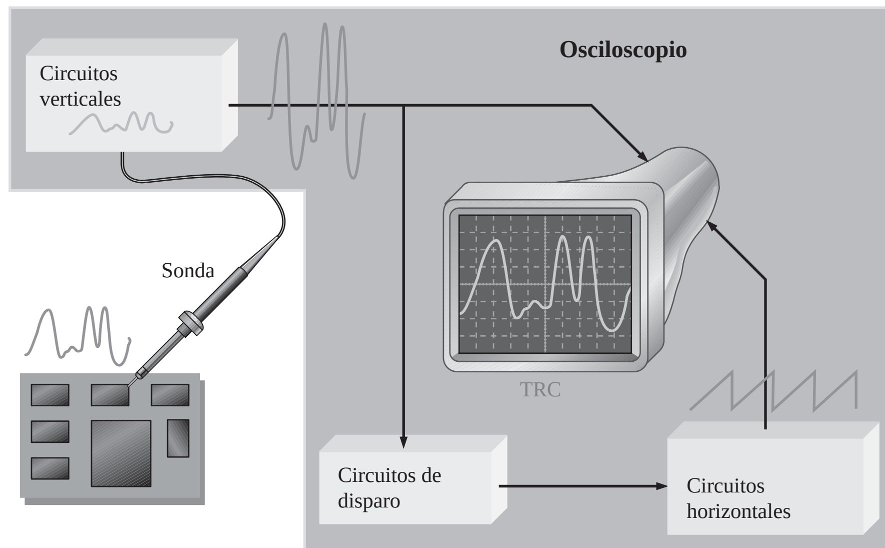


FIGURA 1.42 Diagrama de bloques de un osciloscopio analógico.

Operación básica de los osciloscopios digitales. Algunas partes del osciloscopio digital son similares a las del osciloscopio analógico. Sin embargo, el osciloscopio digital es más complejo que el analógico y, normalmente, dispone de una pantalla LCD en lugar del TRC. En lugar de mostrar una forma de onda tal y como se genera, primero adquiere la forma de onda analógica que se va a medir y la convierte a formato digital utilizando

un convertidor analógico-digital (ADC, *Analog-to-Digital Converter*). Los datos analógicos se almacenan y se procesan. Los datos pasan a continuación a los circuitos de reconstrucción y presentación para poder ser mostrados en la pantalla en su forma original. La Figura 1.43 muestra un diagrama de bloques básico de un osciloscopio digital.

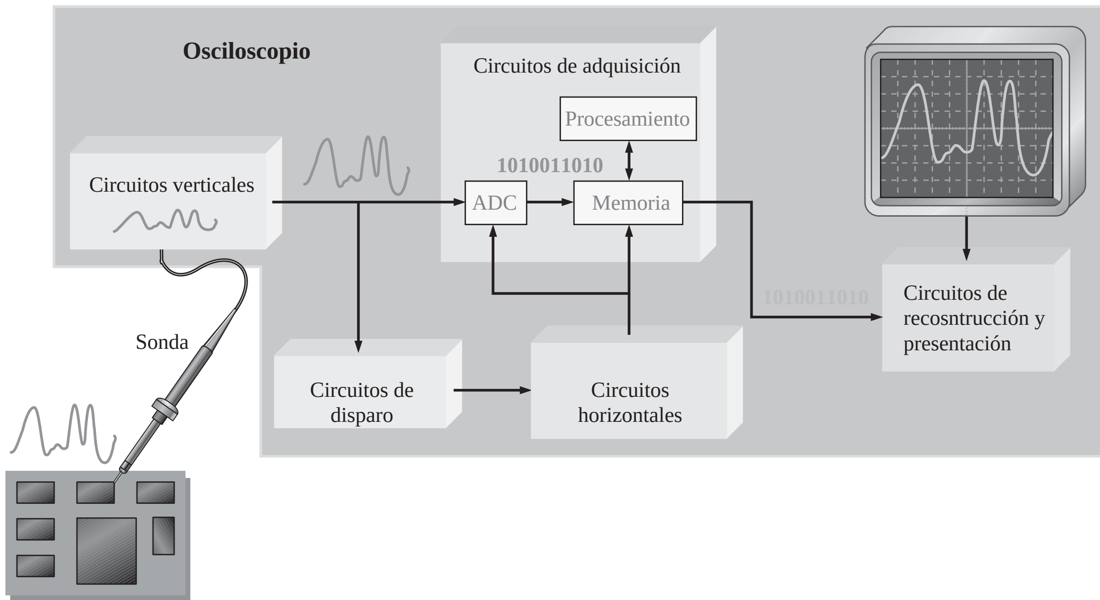


FIGURA 1.43 Diagrama de bloques de un osciloscopio digital.

Controles del osciloscopio. En la Figura 1.44 se muestra el panel frontal de un osciloscopio de doble canal típico. Los instrumentos variarán dependiendo del modelo y del fabricante, aunque la mayoría de ellos tienen determinadas funciones comunes. Por ejemplo, las dos secciones verticales contienen un control de posición (Position), un botón menú de canal y un control V/div (voltios/división). La sección horizontal dispone de un control sec/div (segundos/división). A continuación vamos a explicar algunos de los principales controles del osciloscopio. Para conocer los detalles concretos de un determinado osciloscopio consulte el manual de usuario.

Controles verticales. En la sección vertical del osciloscopio de la Figura 1.44, hay disponibles controles idénticos para cada uno de los dos canales (CH1 y CH2). El control Position permite desplazar la forma de onda mostrada en pantalla en sentido vertical hacia arriba y hacia abajo. El botón Menu permite seleccionar los distintos elementos que aparecerán en pantalla, como por ejemplo los modos de acoplamiento (ac, dc o tierra), el ajuste grueso o fino para el control V/div, la atenuación de la sonda y otros parámetros. El control V/div ajusta el número de voltios representados por cada división vertical de la pantalla. La configuración de V/div para cada canal aparece en la parte inferior de la pantalla. El botón Math Menu proporciona una selección de operaciones que pueden realizarse sobre las formas de onda de entrada, como por ejemplo sustracción, suma o inversión.

Controles horizontales. En la sección horizontal, los controles se aplican a ambos canales. El control Position permite desplazar la forma de onda en sentido horizontal por la pantalla hacia la izquierda o la derecha. El botón Menu permite seleccionar distintos elementos que aparecen en pantalla, como la base de tiempos principal, una vista ampliada de una parte de la señal y otros parámetros. El control sec/div ajusta el tiempo representado por cada división horizontal o base de tiempos principal. La configuración del control sec/div aparecerá en la parte inferior de la pantalla.

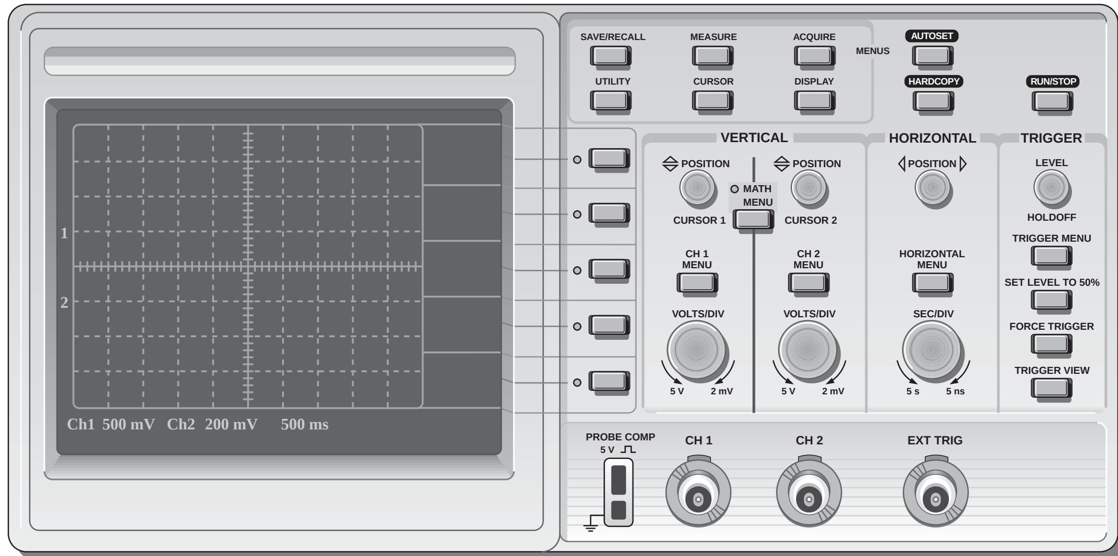
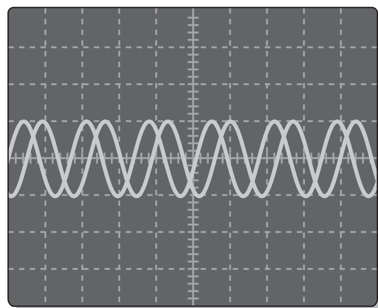


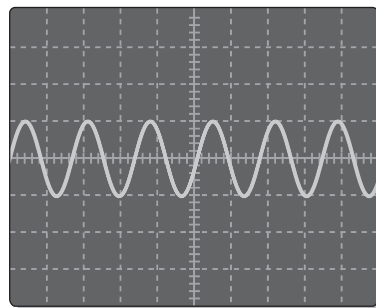
FIGURA 1.44 Osciloscopio típico de doble canal. Los números de la parte inferior de la pantalla indican los valores de cada división de la escala vertical (tensión) y de la escala horizontal (tiempo) y se pueden variar utilizando los controles vertical y horizontal del osciloscopio.

Controles de disparo (trigger). En la sección de controles Trigger, el control Level (nivel) determina el punto de la forma de onda en el se produce el disparo con el fin de iniciar el barrido para visualizar las formas de onda de entrada. El botón Menu permite seleccionar varios elementos que aparecen en pantalla, entre los que se incluyen el flanco o pendiente de disparo, el origen de disparo, el modo de disparo y otros parámetros. Existe también una entrada para la señal externa de disparo. Los controles Trigger estabilizan la forma de onda en la pantalla o generan apropiadamente disparos sobre un impulso que se produce sólo una vez o de forma aleatoria. También permiten observar los retardos de tiempo entre dos señales. La Figura 1.45 compara una señal a la que se la ha aplicado un punto de disparo y otra a la que no. La señal sin punto de disparo tiende a derivar a lo largo de la pantalla, generando lo que parecen múltiples formas de onda.

Acoplamiento de una señal al osciloscopio. El método que se emplea para conectar la señal de tensión que se va a medir al osciloscopio es el acoplamiento. Se suele seleccionar el acoplamiento DC y AC en el



(a) Forma de onda sin disparo



(b) Forma de onda con disparo

FIGURA 1.45 Comparación en un osciloscopio de una señal a la que se la ha aplicado un punto de disparo y otra a la que no.

menú Vertical del osciloscopio. El acoplamiento DC permite visualizar una señal incluyendo su componente continua. El acoplamiento AC bloquea la componente continua de la señal, por lo que la forma de onda se visualiza centrada en 0 V. El modo Ground (tierra) nos permite conectar la entrada del canal a tierra para ver en la pantalla dónde se encuentra la referencia de 0 V. La Figura 1.46 ilustra el resultado de un acoplamiento DC y AC utilizando un tren de impulsos que tiene una componente continua.

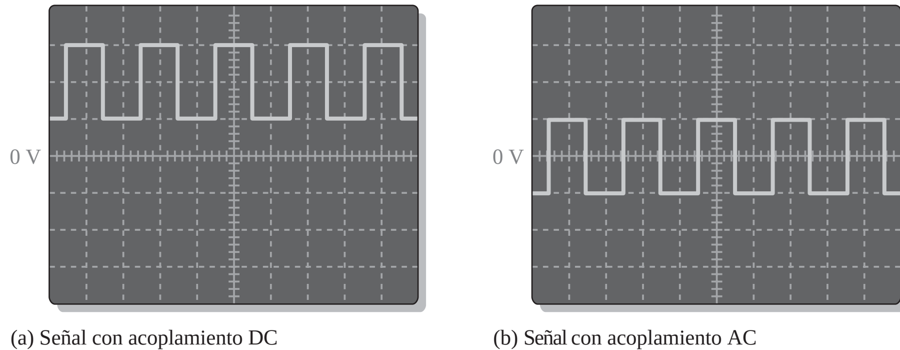


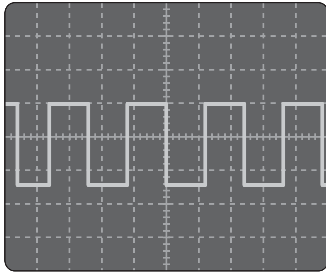
FIGURA 1.46 Visualizaciones de una misma forma de onda con una componente continua.

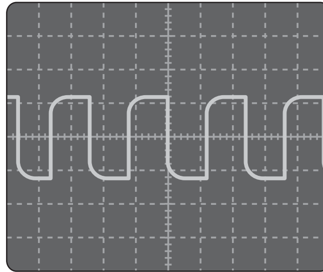
La sonda de tensión mostrada en la Figura 1.47 es esencial para conectar una señal al osciloscopio. Puesto que todos los instrumentos tienden a afectar al circuito que se está midiendo debido a los efectos de carga, la mayoría de las sondas de prueba proporcionan una resistencia serie grande con el fin de minimizar dichos efectos de carga. Las sondas que tienen una resistencia serie diez veces mayor que la resistencia de entrada del osciloscopio se denominan sondas ×10. Las sondas que no presentan resistencia serie se llaman sondas ×1. El osciloscopio ajusta su calibración de acuerdo con la atenuación del tipo de sonda que se vaya a utilizar. Para la mayor parte de medidas, es aconsejable utilizar la sonda ×10. Sin embargo, si se van a medir señales muy pequeñas, una sonda ×1 puede resultar ser una buena elección.



FIGURA 1.47 Sonda de tensión de un osciloscopio. Fotografía utilizada con permiso de Tektronix, Inc.

La sonda dispone de un ajuste que permite compensar la capacitancia de entrada del osciloscopio. La mayoría de los osciloscopios tienen una salida de compensación de sonda que proporciona una onda cuadrada calibrada para llevar a cabo la compensación de la sonda. Antes de realizar la medida, es necesario asegurarse de que la sonda está compensada correctamente para eliminar cualquier distorsión que se haya introducido. Normalmente, hay disponible un tornillo u otro elemento para ajustar la compensación de la sonda. La Figura 1.48 muestra las formas de onda visualizadas en el osciloscopio para las tres condiciones de la sonda:


Correctamente compensada


Subcompensada

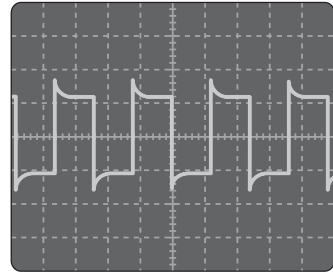

Sobrecompensada

FIGURA 1.48 Condiciones de compensación de una sonda.

EJEMPLO 1.3

Basándose en las lecturas mostradas, determinar la amplitud y el período del tren de pulsos mostrado en la pantalla del osciloscopio de la Figura 1.49. Calcular también la frecuencia.

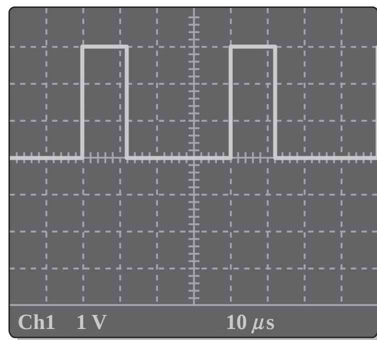


FIGURA 1.49

Solución El control V/div está en 1 V. Los pulsos tienen una altura de tres divisiones. Dado que cada división representa 1 V, la amplitud del pulso es

Amplitud = (3 div)(1 V/div) = **3 V**

El control sec/div se encuentra en 10 μs. Un ciclo completo de la forma de onda (desde el principio de un pulso hasta el principio del siguiente) ocupa cuatro divisiones; por tanto, el período es:

Período (T) = (4 div)(10 μs/div) = **40 μs**

La frecuencia se calcula como sigue

$$f = \frac{1}{T} = \frac{1}{40\mu s} = \mathbf{25\,kHz}$$

Problema relacionado Para V/div igual a 4 V y sec/div en 2 ms, determinar al amplitud y el período del pulso mostrado en la Figura 1.49.

correctamente compensada, subcompensada y sobrecompensada. Si la forma de onda se muestra subcompensada o sobrecompensada, debe ajustarse la sonda hasta conseguir una onda cuadrada compensada correctamente.

El analizador lógico

Los analizadores lógicos se emplean para realizar medidas de múltiples señales digitales y en situaciones en las que los requisitos de disparo sean complejos. Básicamente, el analizador lógico se utiliza para el análisis de circuitos con microprocesadores en los que la localización de averías y los procesos de depuración requieren muchas más entradas que las que ofrece un osciloscopio. Muchos osciloscopios disponen de dos canales de entrada y algunos tienen cuatro. Existen analizadores lógicos que tienen desde 34 hasta 136 canales de entrada. Generalmente, el osciloscopio se utiliza para medir la amplitud, frecuencia y otros parámetros de temporización de unas pocas señales simultáneas o cuando se desean medir parámetros como los tiempos de subida y de bajada, los picos de señal y los retardos. El analizador lógico se emplea cuando es necesario determinar los niveles lógicos de una gran cantidad de señales y para conocer la correlación de señales simultáneas basándose en sus relaciones temporales. En la Figura 1.50 se muestra un analizador lógico típico y en la Figura 1.51 se presenta un diagrama de bloques simplificado.



FIGURA 1.50 Analizador lógico típico. Fotografía utilizada con permiso de Tektronix, Inc.

Adquisición de datos. La gran cantidad de señales que un analizador lógico puede adquirir a un mismo tiempo es uno de los principales factores que le diferencia del osciloscopio. Generalmente, los dos tipos de adquisición de datos de los que dispone un analizador lógico son la adquisición de tiempo y la adquisición de estados. La adquisición de tiempos se emplea fundamentalmente cuando se necesitan determinar las relaciones temporales entre varias señales. La adquisición de estados se utiliza cuando se necesita ver la secuencia de estados que va apareciendo en un sistema bajo prueba.

A menudo resulta útil tener los datos de estado y de temporización correlados, y la mayoría de los analizadores lógicos pueden adquirir simultáneamente dichos datos. Por ejemplo, puede detectarse inicialmente un problema como por ejemplo un estado no válido. Si embargo, la condición de invalidez puede deberse a que se produce una violación de temporización en el sistema bajo prueba. Si no se dispusiera de ambos tipos de información al mismo tiempo, aislar el problema resultaría muy complicado.

Número de canales y profundidad de memoria. Los analizadores lógicos contienen una memoria de adquisición de tiempo real en la que se almacenan los datos muestreados de todos los canales a medida que se producen.

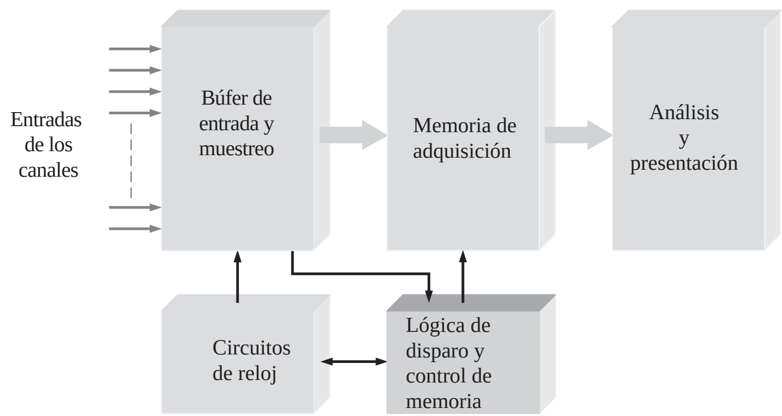
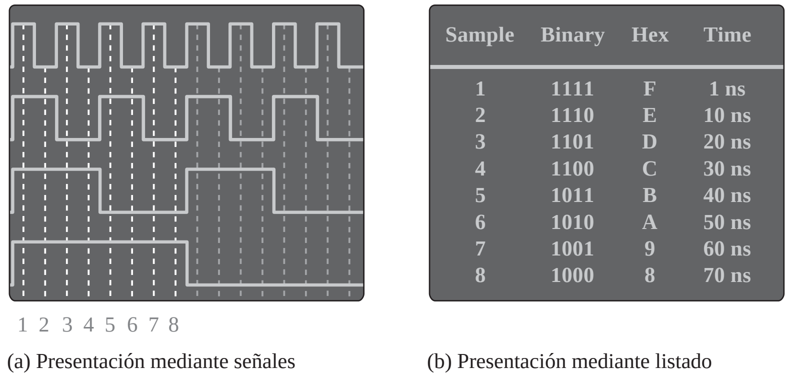


FIGURA 1.51 Diagrama de bloques simplificado de un analizador lógico.

Dos características enormemente importantes son el número de canales y la profundidad de memoria. Puede pensarse en la memoria de adquisición como en una memoria que tiene un ancho igual al número de canales y una profundidad que es el número de bits que cada canal puede capturar durante un determinado intervalo de tiempo.

El número de canales determina el número de señales que se puede adquirir simultáneamente. En ciertos tipos de sistemas hay presentes una gran cantidad de señales, como por ejemplo en el bus de datos de un sistema basado en microprocesador. La profundidad de la memoria de adquisición determina la cantidad de datos procedentes de un determinado canal que se pueden visualizar en cualquier instante de tiempo dado.

Análisis y presentación. Una vez que los datos se han muestreado y almacenado en la memoria de adquisición, suelen utilizarse en varios modos de análisis y presentación diferentes. La forma de onda mostrada es muy similar a la que se puede ver en un osciloscopio en el que se muestren las relaciones temporales de múltiples señales. La pantalla que contiene el listado indica el estado del sistema bajo prueba, especificando los valores de las formas de onda de entrada (1s y 0s) en distintos instantes de tiempo (puntos de muestreo). Normalmente, estos datos pueden presentarse en formato hexadecimal o en otros formatos. La Figura 1.52 ilustra las versiones simplificadas de estos modos de presentación. La presentación en forma de listado especifica los puntos correspondientes a las muestras numeradas en el modo de presentación mediante señales. En el siguiente capítulo se estudian los sistemas de numeración binario y hexadecimal.



Sample	Binary	Hex	Time
1	1111	F	1 ns
2	1110	E	10 ns
3	1101	D	20 ns
4	1100	C	30 ns
5	1011	B	40 ns
6	1010	A	50 ns
7	1001	9	60 ns
8	1000	8	70 ns

(a) Presentación mediante señales (b) Presentación mediante listado

FIGURA 1.52 Los dos modos de presentación del analizador lógico.

Otros dos modos que resultan útiles en la realización de pruebas de computadoras y sistemas basados en microprocesador son el modo de trazado de instrucciones y el de depuración del código fuente. El trazado de instrucciones determina y presenta las instrucciones que se van ejecutando. En este modo, generalmente se muestran los códigos de operación y los mnemónicos de la instrucciones, además de su correspondiente dirección de memoria. Muchos analizadores lógicos también incluyen un modo de depuración del código fuente, que esencialmente permite ver qué es lo que hace realmente el sistema bajo prueba cuando se ejecuta una instrucción de programa.

Sondas. Con los analizadores lógicos se emplean tres tipos básicos de sondas. Una de ellas es una sonda de compresión multicanal que puede conectarse a puntos de una tarjeta de circuito impreso, como se muestra en la Figura 1.53. Otro tipo es la sonda multicanal, similar a la anterior, que se conecta a zócalos montados sobre el circuito impreso. Y la última es la sonda monocanal de mordaza.

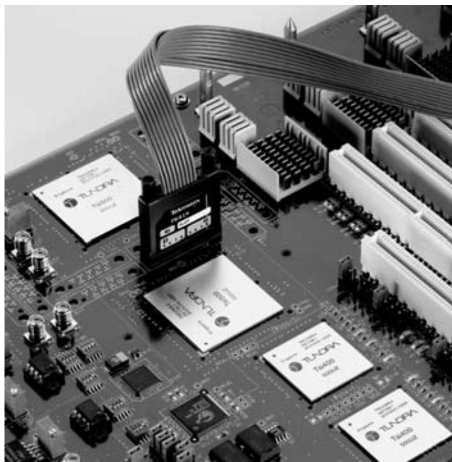


FIGURA 1.53 Una sonda típica de analizador lógico multicanal. Fotografía utilizada con permiso de Tektronix, Inc.

Generadores de señales

Fuente de señal lógica. Estos instrumentos se conocen también como generadores de impulsos y generadores de patrones. Están diseñados específicamente para generar señales digitales con amplitudes y flancos precisos y para generar los flujos de 1s y 0s necesarios para probar los buses de las computadoras, microprocesadores y otros sistemas digitales.

Generadores de señales arbitrarias y generadores de funciones. El generador de señales arbitrarias puede utilizarse para generar señales estándar como ondas sinusoidales, ondas triangulares e impulsos, así como señales con distintas formas y características. Las formas de onda pueden definirse mediante entradas en formato matemático o en formato gráfico. En la Figura 1.54(a) se muestra un generador de señales arbitrarias.

El generador de funciones proporciona trenes de impulsos, así como ondas sinusoidales y triangulares. La mayoría de los generadores de funciones disponen de salidas compatibles lógicas para proporcionar los niveles apropiados de excitación de las entradas de los circuitos digitales. En la Figura 1.54(b) se muestran varios generadores de funciones típicos.

La sonda lógica y el pulsador lógico. La sonda lógica es una herramienta muy útil y barata que proporciona un medio para la localización de averías en un circuito digital, detectando las condiciones en un punto del circuito, como se ilustra en la Figura 1.55. La sonda puede detectar niveles de tensión altos, niveles de tensión



(a) Un generador de señales arbitrario. (b) Ejemplos de generadores de funciones.

FIGURA 1.54 Generadores de señales típicos. Fotografías utilizadas con permiso de Tektronix, Inc.

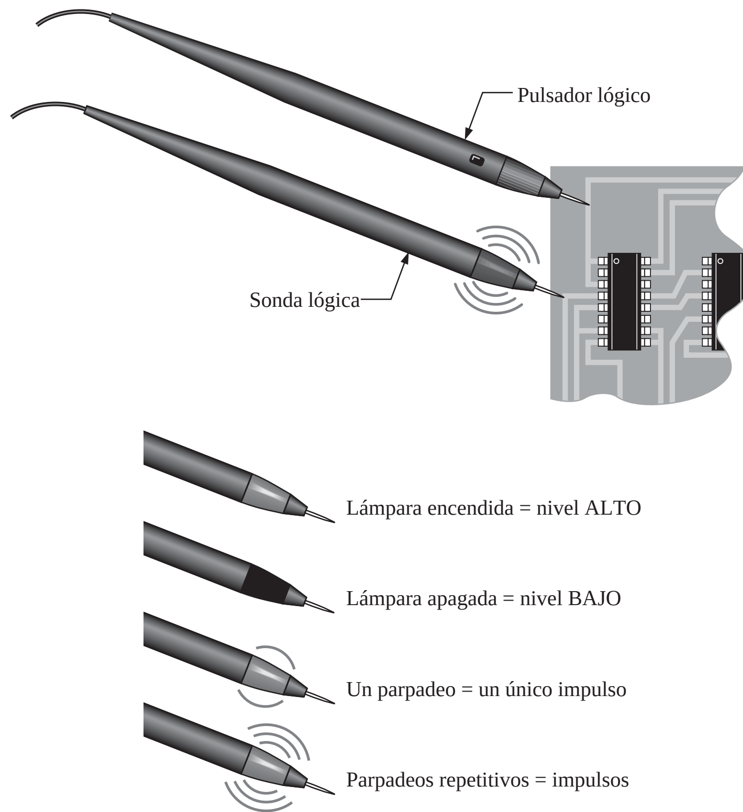


FIGURA 1.55 Ilustración de cómo se pueden utilizar un pulsador lógico y una sonda lógica para aplicar un impulso en un determinado punto y comprobar la actividad resultante del impulso en otra parte del circuito.

bajos, impulsos aislados, impulsos repetitivos y circuitos abiertos en una tarjeta de circuito impreso. La luz de la sonda indica la condición que existe en un determinado punto del circuito, tal y como se indica en la figura. El pulsador lógico genera un tren de pulsos repetitivos que se pueden aplicar a cualquier punto del circuito. Se pueden aplicar impulsos en un punto de un circuito con el pulsador y comprobar en algún otro punto los impulsos resultantes con una sonda lógica.

Otros instrumentos

Fuente de alimentación continua. Este instrumento es indispensable en cualquier banco de pruebas. La fuente de alimentación convierte la alimentación alterna que se obtiene de la red eléctrica en una tensión continua regulada. Todos los circuitos digitales necesitan tensión continua. Muchos circuitos lógicos requieren +5 V o + 3,3 V para funcionar. La fuente de alimentación se utiliza para alimentar a los circuitos durante las fases de diseño, de desarrollo y para la localización de averías cuando la alimentación del sistema no está disponible. En la Figura 1.56 se muestran fuentes de alimentación continua típicas para bancos de pruebas.



FIGURA 1.56 Fuentes de alimentación continua típicas. Cortesía de B +K Precision.®

Multímetro digital. El multímetro digital se usa para medir tensiones continuas, tensiones alternas, resistencias, etc. La Figura 1.57 muestra dos típicos multímetros de sobremesa y portátil.



FIGURA 1.57 Multímetros digitales típicos. Cortesía de B + K Precision.®

REVISIÓN DE LA SECCIÓN 1.7

1. ¿Cuál es la diferencia principal entre un osciloscopio analógico y un osciloscopio digital?
2. Cite las dos diferencias principales entre un analizador lógico y un osciloscopio.
3. ¿Para qué sirve el control V/div de un osciloscopio?
4. ¿Para qué sirve el control sec/div de un osciloscopio?
5. ¿Cuál es el propósito de un generador de funciones?



APLICACIÓN A LOS SISTEMAS DIGITALES

En esta sección se presenta una aplicación simplificada de un sistema que contiene los elementos y funciones lógicas que se han explicado en la Sección 1.4. Es importante que el lector comprenda cómo varias funciones digitales pueden operar conjuntamente como un sistema completo para llevar a cabo una tarea específica. También es importante empezar a pensar en términos de funcionamiento a nivel de sistema ya que, en la práctica, gran parte de su trabajo va a implicar el tratar con sistemas y no con funciones individuales. Por supuesto, para comprender los sistemas, primero se deben entender las funciones y elementos básicos que los conforman.

Esta sección introduce el concepto de sistema. El ejemplo le mostrará cómo pueden operar conjuntamente las funciones lógicas para realizar una tarea de alto nivel, y le permitirá comenzar a pensar a nivel de sistema. El sistema específico que se va utilizar aquí para ilustrar el concepto de sistema sirve como modelo y no es necesariamente el método que se usaría en la práctica, aunque podría serlo. En las aplicaciones industriales modernas, como la que se trata aquí, a menudo se emplean ciertos instrumentos conocidos como controladores programables.

El sistema

Imaginemos que una empresa utiliza el sistema de control de procesos mostrado en el diagrama de bloques simplificado de la Figura 1.58, para contar y envasar automáticamente pastillas. Las pastillas se introducen en un gran embudo conductor. El estrecho cuello del embudo sólo permite que caiga una pastilla dentro del bote que se encuentra sobre la cinta transportadora.

El sistema digital controla el número de pastillas que caen en cada bote y muestra el total de pastillas actualizado continuamente en una pantalla próxima a la línea de montaje, así como en una localización remota situada en otra parte de la planta de la fábrica. Este sistema utiliza todas las funciones lógicas básicas presentadas en la Sección 1.4 y su único propósito es el de mostrar cómo pueden combinarse estas funciones para alcanzar el resultado deseado.

El funcionamiento general es el siguiente: un sensor óptico en la parte inferior del embudo detecta cada pastilla que pasa y genera un impulso eléctrico. Este impulso llega al contador y hace que éste se incremente en una unidad, por lo que mientras que el bote se está llenando, el contador almacena la representación binaria del número de pastillas que hay en el bote. Este número binario se transfiere desde el contador a través de las líneas paralelo hasta la entrada *B* del comparador. En la entrada *A* del comparador se aplica el número binario preseleccionado igual a la cantidad de pastillas que hay que introducir en cada bote. Este número preseleccionado procede del teclado numérico y de su circuitería asociada, que incluye el codificador, el registro A y el convertidor de código A. Cuando el número deseado de pastillas se introduce en el teclado, se codifica y se almacena en el registro paralelo A hasta que se requiere cambiar la cantidad de pastillas por bote.

Por ejemplo, supongamos que cada bote va a contener 50 pastillas. Cuando el número del contador alcanza este valor, la salida $A = B$ del comparador pasa a nivel ALTO, lo que indica que el bote está lleno.

La salida a nivel ALTO del comparador cierra inmediatamente la válvula del cuello del embudo para detener el flujo de pastillas y, al mismo tiempo, activa la cinta transportadora para mover el siguiente bote y situarlo debajo del embudo. Cuando el siguiente bote está correctamente colocado debajo del cuello del embudo, el circuito de control de la cinta transportadora genera un impulso que pone a cero el contador. La salida $A = B$ del comparador pasa a nivel BAJO, abriendo la válvula del embudo para reiniciar el flujo de pastillas.

En la parte del sistema correspondiente al display, el número contenido en el contador se transmite en paralelo a la entrada *A* del sumador. La entrada *B* del mismo procede del registro B que almacena el número total de pastillas envasadas, hasta el último bote que se ha llenado. Por ejemplo, si se han llenado diez botes y cada bote contenía cincuenta pastillas, el registro B contiene la representación binario del número 500. A continuación, cuando se ha rellenado el siguiente bote, el número binario correspondiente a 50 aparecerá en la entrada *A* del sumador y en la entrada *B* estará el número binario correspondiente a 500. El sumador genera una nueva suma, cuyo resultado es 550 que se almacena en el registro B, reemplazando a la suma anterior de 500.

El número binario contenido en el registro B se transmite en paralelo al convertidor de código y al decodificador, que lo pasa de formato binario a decimal para mostrarlo en el display o en alguna pantalla próxima a la cinta transportadora. El contenido del registro se transmite también a un multiplexor, de forma que pueda ser convertido de paralelo a serie para ser transmitido por una única línea hasta la ubicación remota a cierta distancia. Es más económico instalar una única línea cuando la distancia es relativamente grande y la velocidad de los datos no es un factor

importante en la aplicación que varias líneas para transmitir en paralelo. En la localización remota, los datos serie se demultiplexan y se envían al registro C. A partir de este punto, los datos se decodifican para poder mostrarlos en la pantalla remota.

Recuerde que este sistema es puramente un modelo ilustrativo y no necesariamente representa ni el más actual ni el más eficiente método para implementar este hipotético proceso. Aunque efectivamente existen muchos otros métodos, se ha seleccionado éste en particular para poder ilustrar todas las funciones lógicas que se han introducido en la Sección 1.4, y que se estudiarán en detalle en los próximos capítulos. Este ejemplo muestra una aplicación de los distintos dispositivos funcionales en el nivel del sistema y cómo pueden conectarse para alcanzar un objetivo específico.

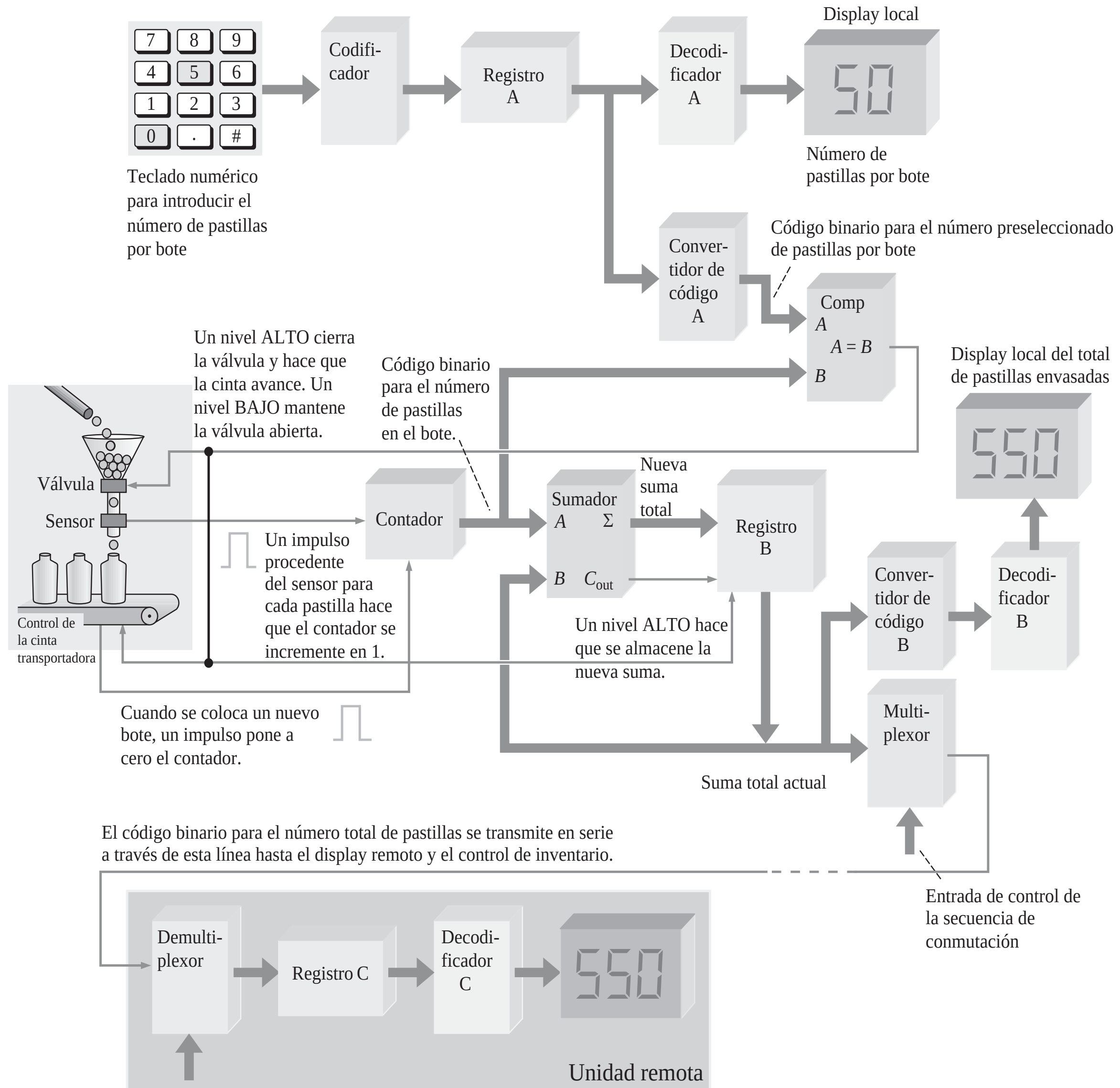


FIGURA 1.58 Diagrama de bloques simplificado de un sistema de control de recuento y envasado de pastillas.

RESUMEN

- Una magnitud analógica posee un conjunto continuo de valores.
- Una magnitud digital posee un conjunto discreto de valores.
- Un dígito binario se denomina bit.
- Un impulso se caracteriza por el tiempo de subida, el tiempo de bajada, la anchura del impulso y la amplitud.
- La frecuencia de una señal periódica es el recíproco de su período. Las fórmulas que relacionan la frecuencia y el período son:

$$f = \frac{1}{T} \quad \text{y} \quad T = \frac{1}{f}$$

- El ciclo de trabajo de un tren de pulsos es la relación entre el ancho del impulso y el período, expresado como un porcentaje según la siguiente fórmula:

$$\text{Ciclo de trabajo} = \left(\frac{t_W}{T}\right)100\%$$

- Un diagrama de tiempos es una representación de dos o más formas de onda que muestra su relación con respecto al tiempo.
- Las tres operaciones lógicas básicas son NOT, AND y OR. Sus símbolos estándar son los indicados en la Figura 1.59.

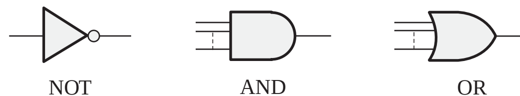


FIGURA 1.59

- Las funciones lógicas básicas son: comparación, aritmética, conversión de código, decodificación, codificación, selección de datos, almacenamiento y recuento.
- Las dos categorías físicas más importantes de los encapsulados de CI son: montaje de inserción y montaje superficial.
- Las categorías de los CI según la complejidad del circuito son: SSI (integración a baja escala), MSI (integración a media escala), LSI, VLSI y ULSI (integración a gran escala, a muy gran escala y a ultra gran escala).
- Los dos tipos de dispositivos lógicos programables simples (SPLD) son: PAL (*Programmable Array Logic*) y GAL (*Generic Array Logic*).
- El CPLD (*Complex Programmable Logic Device*) contiene múltiples SPLD con interconexiones programables.
- La FPGA (*Field Programmable Gate Arra*y) tiene una estructura interna diferente que el CPLD y, generalmente, se utiliza para circuitos y sistemas más complejos.
- Los instrumentos más comúnmente utilizados para la realización de pruebas y la localización de averías de los circuitos digitales son: el osciloscopio, el analizador lógico, el generador de señales, el generador de funciones, la fuente de alimentación continua, el multímetro digital, la sonda lógica y el pulsador lógico.

PALABRAS CLAVE

Las palabras clave y otros términos que se han resaltado en negrita se encuentran también en el glosario final del libro.

Analógico Que es continuo o toma valores continuos.

AND Operación lógica básica en la que se obtiene una salida verdadera (nivel ALTO) sólo cuando todas las condiciones de entrada son verdaderas (nivel ALTO).
Binario Que tiene dos valores o estados; describe un sistema de numeración en base 2 y utiliza como dígitos 1 y 0.
Bit Dígito binario, que puede ser 1 o 0.
Circuito integrado (CI) Un tipo de circuito en el que todos los componentes están integrados en un solo chip de material semiconductor de tamaño extremadamente pequeño.
Compilador Programa que controla el flujo de diseño y traduce el código fuente en código objeto en un formato que puede ser probado lógicamente y descargado en un dispositivo objetivo.
CPLD *Complex Programmable Logic Device.* Un dispositivo lógico programable complejo que consta básicamente de múltiples matrices SPLD con interconexiones programables.
Datos Información en formato numérico, alfabético o cualquier otro.
Diagrama de tiempos Una gráfica de señales digitales que muestra las relaciones temporales de dos o más formas de onda.
Digital Relativo a los dígitos o magnitudes discretas; que toma un conjunto de valores discretos.
Entrada La señal o línea que entra en un circuito.
FPGA *Field Programmable Gate Array* (matriz de puertas programable por campo).
Impulso Cambio repentino de un nivel a otro, seguido, tras un cierto tiempo, denominado ancho del impulso, de otro cambio repentino al nivel original.
Inversor Circuito NOT; un circuito que cambia un nivel ALTO a un nivel BAJO, o viceversa.
Localización de averías La técnica o proceso de identificar, aislar y corregir de forma sistemática una fallo en un circuito o sistema.
Lógica En electrónica digital, la capacidad de toma de decisiones de los circuitos de puertas, en los que un nivel ALTO representa una sentencia verdadera y un nivel BAJO representa una sentencia falsa.
NOT Operación lógica básica que realiza operaciones de inversión.
OR Operación lógica básica en la que una salida verdadera (nivel ALTO) se produce cuando una o más de las condiciones entrada son verdaderas (nivel ALTO).
Paralelo En los sistemas digitales, datos que se generan simultáneamente en varias líneas; la transferencia o procesamiento simultáneo de varios bits.
Puerta Circuito lógico que realiza una operación lógica especificada, como por ejemplo AND u OR.
Reloj Señal de temporización básica en un sistema digital; una señal periódica en la que cada intervalo entre impulsos es igual a la duración de un bit.
Salida Señal o línea que sale de un circuito.
Serie Disponer elementos uno detrás de otro, como en una transferencia serie de bits; ocurrencia en secuencia en lugar de simultáneamente.
SPLD *Simple Programmable Logic Device* (dispositivo lógico programable simple).

AUTOTEST

Las respuestas se encuentran al final del capítulo.

1. Una magnitud que toma valores continuos es:

(a) una magnitud digital **(b)** una magnitud analógica
(c) un número binario **(d)** un número natural

2. El término *bit* significa:

(a) una pequeña cantidad de datos **(b)** un 1 o un 0
(c) dígito binario **(d)** las respuestas (b) y (c)

3. El intervalo de tiempo en el flanco anterior de un impulso entre el 10% y el 90% de la amplitud es el:

(a) tiempo de subida **(b)** tiempo de bajada **(c)** ancho del impulso **(d)** período

4. Un impulso de una cierta forma de onda se produce cada. La frecuencia es:
(a) 1 kHz **(b)** 1 Hz **(c)** 100 Hz **(d)** 10 Hz

5. En una determinada señal digital, el período es dos veces el ancho del impulso. El ciclo de trabajo es:
(a) 100% **(b)** 200% **(c)** 50%

6. Un inversor
(a) realiza la operación NOT **(b)** cambia de un nivel ALTO a un nivel BAJO
(c) cambia de un nivel BAJO a un nivel ALTO **(d)** todas las anteriores

7. La salida de una puerta AND es un nivel ALTO cuando
(a) cualquier entrada está a nivel ALTO **(b)** todas las entradas están a nivel ALTO
(c) ninguna entrada está a nivel ALTO **(d)** las respuestas (a) y (b)

8. La salida de una puerta OR está a nivel ALTO cuando
(a) cualquier entrada está a nivel ALTO **(b)** todas las entradas están a nivel ALTO
(c) ninguna entrada está a nivel ALTO **(d)** las respuestas (a) y (b)

9. El dispositivo utilizado para convertir un número binario en un formato para display de 7-segmentos es el:
(a) multiplexor **(b)** codificador **(c)** decodificador **(d)** registro

10. Un ejemplo de un dispositivo de almacenamiento de datos es:
(a) la puerta lógica **(b)** el flip-flop
(c) el comparador **(d)** el registro **(e)** las respuestas (b) y (d)

11. Un encapsulado de CI de función fija que contiene cuatro puertas AND es un ejemplo de:
(a) MSI **(b)** SMT **(c)** SOIC **(d)** SSI

12. Un dispositivo LSI tiene una complejidad de
(a) 10 a 100 puertas equivalentes **(b)** más de 100 a 10.000 puertas equivalentes
(c) 2000 a 5000 puertas equivalentes **(d)** más de 10.000 a 100.000 puertas equivalentes

13. VHDL es un
(a) dispositivo lógico **(b)** lenguaje de programación de dispositivos PLD
(c) lenguaje de computadora **(d)** *Very High Density Logic*

14. Un CPLD es un
(a) controlled program logic device **(b)** complex programmable logic driver
(c) complex programmable logic device **(d)** central processing logic device

15. Una FPGA es una
(a) field programmable gate array **(b)** fast programmable gate array
(c) field programmable generic array **(d)** flash process gate application

PROBLEMAS

Las respuestas a los problemas impares se encuentran al final del libro.

SECCIÓN 1.1 Magnitudes analógicas y digitales

1. Nombre dos ventajas de los datos digitales en comparación con los datos analógicos.
2. Nombre una magnitud analógica distinta de la temperatura o del sonido.

SECCIÓN 1.2 Dígitos binarios, niveles lógicos y formas de onda digitales

3. Defina la secuencia de bits (1s y 0s) representada por cada una de estas secuencias de niveles:
   (a) ALTO, ALTO, BAJO, ALTO, BAJO, BAJO, BAJO, ALTO
   (b) BAJO, BAJO, BAJO, ALTO, BAJO, ALTO, BAJO, ALTO, BAJO
4. Enumere la secuencia de niveles (ALTO y BAJO) que representa cada una de las siguientes secuencias de bits:
   (a) 1 0 1 1 1 0 1 **(b)** 1 1 1 0 1 0 0 1
5. Para el impulso mostrado en la Figura 1.60, determinar gráficamente los parámetros:
   (a) tiempo de subida **(b)** tiempo de bajada **(c)** ancho del impulso **(d)** amplitud

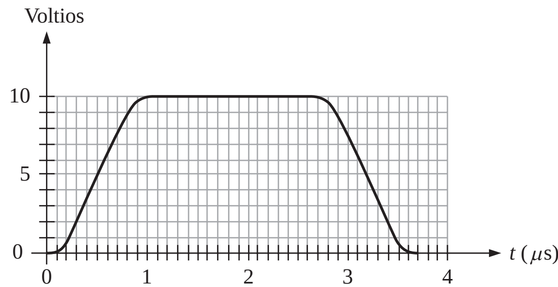


FIGURA 1.60

6. Determinar el período de la forma de onda digital de la Figura 1.61.
7. ¿Cuál es la frecuencia de la forma de onda de la Figura 1.61?

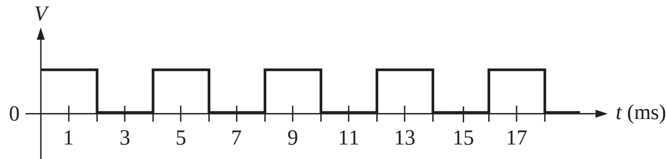


FIGURA 1.61

8. El tren de impulsos de la Figura 1.61 ¿es periódico o no periódico?
9. Determinar el ciclo de trabajo de la forma de onda de la Figura 1.61.
10. Determinar la secuencia de bits representada por la forma de onda de la Figura 1.62. En este caso, un tiempo de bit es igual a 1 μs.
11. ¿Cuál es el tiempo total de transferencia serie para los ocho bits de la Figura 1.62. ¿Cuál es el tiempo total de transferencia en paralelo?

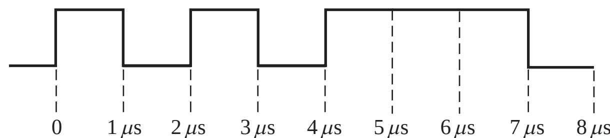


FIGURA 1.62

SECCIÓN 1.3 Operaciones lógicas básicas

12. Un circuito lógico requiere un nivel ALTO en todas sus entradas para poner su salida a nivel ALTO. ¿De qué tipo de circuito lógico se trata?

13. Un circuito lógico básico de 2 entradas tiene un nivel ALTO en una entrada y un nivel BAJO en la otra entrada, y la salida está a nivel BAJO. Identificar el circuito.

14. Un circuito lógico básico de 2 entradas tiene un nivel ALTO en una entrada y un nivel BAJO en la otra entrada, y la salida está a nivel ALTO. ¿De qué circuito lógico se trata?

SECCIÓN 1.4 Introducción a las funciones lógicas básicas

15. Nombre las funciones lógicas de cada bloque de la Figura 1.63 basándose en la observación de las entradas y de las salidas.

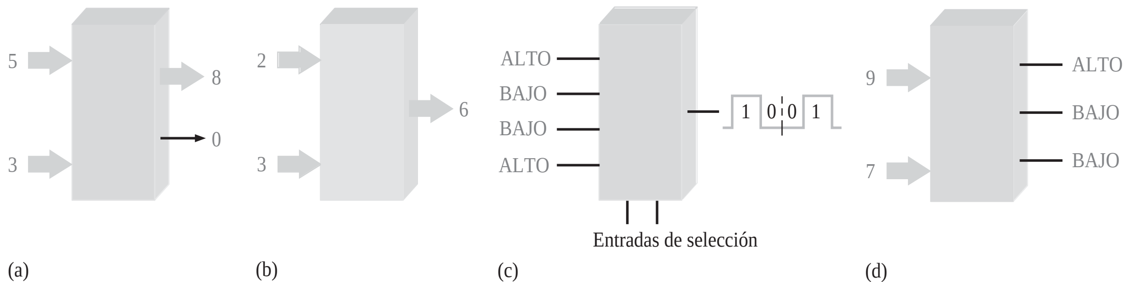


FIGURA 1.63

16. Un tren de impulsos con una frecuencia de 10 kHz se aplica a la entrada de un contador. En 100 ms, ¿cuántos pulsos se contarán?

17. Considere un registro que puede almacenar ocho bits. Suponga que se ha puesto a cero para que contenga ceros en todas sus posiciones. Si transferimos cuatro bits alternativos (0101) en serie al registro, empezando con un 1 y desplazándolo hacia la derecha, ¿cuál será el contenido total del registro cuando se almacena el cuarto bit?

SECCIÓN 1.5 Circuitos integrados de función fija

18. Un CI digital de función fija tiene una complejidad de 200 puertas equivalente. ¿Cómo se clasificaría?

19. Explique la principal diferencia entre los encapsulados DIP y SMT.

20. Numere los pines de los encapsulados mostrados en la Figura 1.64. Se muestran las vistas frontales.

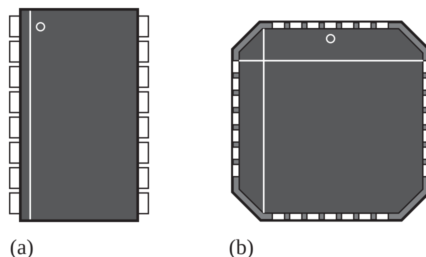


FIGURA 1.64

SECCIÓN 1.6 Introducción a la lógica programable

21. ¿Cuál de los siguientes acrónimos no describe un dispositivos lógico programable?

PAL, GAL, SPLD, ABEL, CPLD, CUPL, FPGA

22. ¿Para qué sirven cada uno de los siguientes dispositivos?

(a) SPLD **(b)** CPLD **(c)** HDL **(d)** FPGA **(e)** GAL

23. Defina cada uno de los siguientes términos de programación de dispositivos PLD:

(a) introducción del diseño **(b)** simulación **(c)** compilación **(d)** descarga

24. Describa el proceso de colocación y rutado.

SECCIÓN 1.7 Instrumentos de medida y prueba

25. Se visualiza un impulso en la pantalla de un osciloscopio y se mide la línea base en 1 V y el máximo del impulso está en 8 V. ¿Cuál es la amplitud?

26. Se aplica una sonda lógica a un punto de contacto en un CI que está operando en un sistema. La lámpara de la sonda parpadea repetidamente. ¿Qué es lo que indica?

SECCIÓN 1.8 Aplicación a los sistemas digitales

27. Defina el término *sistema.*

28. En el sistema de la Figura 1.58, ¿por qué son necesarios el multiplexor y el demultiplexor?

29. ¿Qué acción puede tomarse para cambiar el número de pastillas por bote en el sistema de la Figura 1.58?

RESPUESTAS

REVISIONES DE CADA SECCIÓN

SECCIÓN 1.1 Magnitudes analógicas y digitales

1. *Analógico* significa continuo.
2. *Digital* significa discreto.
3. Una magnitud digital posee un conjunto discreto de valores y una magnitud analógica posee un conjunto de valores continuos.
4. Un altavoz es analógico. Un reproductor de CD es analógico y digital. Una computadora es completamente digital.

SECCIÓN 1.2 Dígitos binarios, niveles lógicos y formas de onda digitales

1. Binario significa que tiene dos estados o valores.
2. Un bit es un dígito binario.
3. Los bits son 1 y 0.
4. Tiempo de subida: del 10% al 90% de la amplitud. Tiempo de bajada: del 90% al 10% de la amplitud.
5. La frecuencia es el recíproco del período.
6. Una señal de reloj es la señal de temporización básica de que se derivan otras formas de onda.
7. Un diagrama de tiempos muestra la relación temporal entre dos o más formas de onda.
8. La transferencia en paralelo es más rápida que la transferencia serie.

SECCIÓN 1.3 Operaciones lógicas básicas

1. Cuando la entrada está a nivel BAJO
2. Cuando todas las entradas están a nivel ALTO
3. Cuando cualquier entrada o todas ellas están a nivel ALTO
4. Un inversor es un circuito NOT.
5. Una puerta lógica es un circuito que realiza una operación lógica (AND, OR).

SECCIÓN 1.4 Introducción a las funciones lógicas básicas

1. Un comparador compara las magnitudes de dos números de entrada.
2. Sumar, restar, multiplicar y dividir.

3. Codificar es cambiar un formato familiar como el decimal a un formato codificado como por ejemplo el binario.
4. Decodificar es cambiar un código a un formato familiar; por ejemplo, de binario a decimal.
5. Multiplexar es poner datos procedentes de varias fuentes en una misma línea. Demultiplexar es tomar datos de una sola línea y distribuirlos a muchos destinos.
6. Flip-flops, registros, memorias semiconductoras, discos magnéticos
7. Un contador cuenta sucesos mediante una secuencia de estados binarios.

SECCIÓN 1.5 Circuitos integrados de función fija

1. Un CI es un circuito electrónico que tiene todos sus componentes integrados en un único chip de silicio.
2. DIP: encapsulado dual in; SMT: tecnología de montaje superficial; SOIC: CI de perfil bajo; SSI: integración a baja escala; MSI: integración a media escala; LSI: integración a gran escala; VLSI: integración a muy gran escala; ULSI: integración a ultra gran escala.
3. **(a)** SSI **(b)** MSI **(c)** LSI **(d)** VLSI **(e)** ULSI

SECCIÓN 1.6 Introducción a la lógica programable

1. SPLD (Simple programmable logic device, dispositivo lógico programable simple), CPLD (complex programmable logic device, dispositivo lógico programable complejo) y FPGA (field programmable gate array, matriz de puertas programable por campo).
2. Un CPLD está formado por múltiples SPLD.
3. Introducción del diseño, simulación funcional, síntesis, implementación, simulación de la temporización y descarga.
4. *Introducción del diseño:* el diseño lógico se introduce utilizando software de desarrollo. *Simulación funcional:* el diseño se simula por software para garantizar que funciona lógicamente. *Síntesis:* el diseño se traduce en una lista de componentes (*netlist*). *Implementación:* la lógica desarrollada mediante la *netlist* se mapea en el dispositivo programable. *Simulación de temporización:* el diseño se simula por software para confirmar que no existen problemas de temporización. *Descarga:* el diseño se introduce en el dispositivo programable.

SECCIÓN 1.7 Instrumentos de medida y prueba

1. El osciloscopio analógico aplica la señal que se va a medir directamente a los circuitos de la pantalla. El osciloscopio digital primero convierte la señal que se va a medir a formato digital.
2. El analizador lógico tiene más canales que el osciloscopio y dispone de más de un formato de visualización de los datos.
3. El control V/div define la tensión para cada división de la pantalla.
4. El control sec/div define el tiempo para cada división de la pantalla.
5. El generador de funciones genera varios tipos de formas de onda.

PROBLEMAS RELACIONADOS

1.1 f = 6,67 kHz; Ciclo de trabajo = 16,7%

1.2 Transferencia paralelo: 100 ns; Transferencia serie: 1,6 μs

1.3 Amplitud = 12 V; T = 8 ms

AUTOTEST

1. (b) **2.** (d) **3.** (a) **4.** (c) **5.** (c) **6.** (d) **7.** (b) **8.** (d) **9.** (c) **10.** (e) **11.** (d) **12.** (d) **13.** (b) **14.** (c) **15.** (a)

2

SISTEMAS DE NUMERACIÓN, OPERACIONES Y CÓDIGOS

CONTENIDO DEL CAPÍTULO



OBJETIVOS DEL CAPÍTULO

- Revisión del sistema de numeración decimal.
- Contar en el sistema de numeración binario.
- Convertir de decimal a binario y de binario a decimal.
- Aplicar las operaciones aritméticas a los números binarios.
- Determinar el complemento a 1 y el complemento a 2 de un número binario.
- Expresar los números con signo en los formatos binarios de signo-magnitud, complemento a 1, complemento a 2 y coma flotante.
- Realizar operaciones aritméticas con números binarios con signo.
- Conversión entre los sistemas de numeración binario y hexadecimal.
- Sumar números en formato hexadecimal.
- Conversión entre los sistemas de numeración binario y octal.
- Expresar los números decimales en formato BCD (*Binary Coded Decimal*).
- Sumar números en BCD.
- Conversión entre el sistema binario y el código Gray.
- Interpretar el código ASCII (*American Standard Code for Information Interchang*e).
- Explicar cómo detectar y corregir los errores de código.

PALABRAS CLAVE

- LSB
- MSB
- Byte
- Números en coma flotante
- Hexadecimal
- Octal
- BCD
- Alfanumérico
- ASCII
- Paridad
- Código Hamming

INTRODUCCIÓN

El sistema de numeración binario y los códigos digitales son fundamentales en las computadoras y, en general, en la electrónica digital. Este capítulo está enfocado principalmente al sistema de numeración binario y sus relaciones con otros sistemas de numeración tales como el decimal, hexadecimal y octal. Se cubren las operaciones aritméticas con números binarios con el fin de proporcionar una base para entender cómo trabajan las computadoras y muchos otros tipos de sistemas digitales. También se abordan códigos digitales como el código decimal binario (BCD, *Binary Coded Decimal*), el código Gray y el ASCII. Se presenta el método de paridad para la detección de errores en los códigos y se describe un método para corregir dichos errores. Las explicaciones sobre el uso de la calculadora en determinadas operaciones están basadas en la calculadora gráfica TI-86 y en la calculadora TI-36X. Los procedimientos mostrados pueden variar en otros tipos de calculadoras.

2.1 NÚMEROS DECIMALES

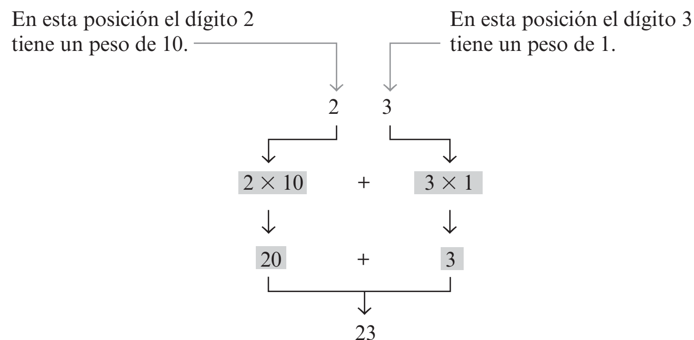
Todos estamos familiarizados con el sistema de numeración decimal porque utilizamos los números decimales todos los días. Aunque los números decimales son triviales, a menudo, su estructura de pesos no se comprende. En esta sección, vamos a repasar la estructura de los números decimales. Este repaso le ayudará a entender más fácilmente la estructura del sistema de numeración binario, que es tan importante en las computadoras y en la electrónica digital.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Explicar por qué el sistema de numeración decimal es un sistema de pesos. ■ Explicar cómo se utilizan las potencias de diez en el sistema decimal. ■ Determinar el peso de cada dígito en un número decimal.

En el sistema de numeración **decimal** cada uno de los diez dígitos, de 0 a 9, representa una determinada cantidad. Como ya sabe, los diez símbolos (**dígitos**) no se limitan a expresar solamente diez cantidades diferentes, ya que usamos varios dígitos en las posiciones adecuadas dentro de un número para indicar la magnitud de la cantidad. Es posible especificar cantidades hasta nueve antes de quedarse sin dígitos; si se desea especificar una cantidad mayor que nueve, se emplean dos o más dígitos y la posición de cada dígito dentro del número indica la magnitud que representa. Por ejemplo, si deseamos expresar la cantidad veintitrés, usaremos (en sus respectivas posiciones dentro del número) el dígito 2 para representar la cantidad de veinte y el dígito 3 para representar la cantidad de 3, como se ilustra a continuación:

▲ *El sistema de numeración decimal utiliza diez dígitos.*



La posición de cada dígito en un número decimal indica la magnitud de la cantidad representada y se le puede asignar un **peso**. Los pesos para los número enteros son las potencias positivas de diez, que aumentan de derecha a izquierda, comenzado por $10^0 = 1$.

▲ *El sistema de numeración decimal es un sistema en base 10.*

$$. . . 10^5\ 10^4\ 10^3\ 10^2\ 10^1\ 10^0$$

Para números fraccionarios, los pesos son las potencias negativas de diez que decrecen de izquierda a derecha comenzando por 10^{-1}.

$$10^2\ 10^1\ 10^0{,}10^{-1}\ 10^{-2}\ 10^{-3} . . .$$

Coma decimal

▲ *El valor de un dígito se determina por su posición dentro del número*

El valor de un número decimal es la suma de los dígitos después de haber multiplicado cada dígito por su peso, como ilustran los Ejemplos 2.1 y 2.2.

EJEMPLO 2.1

Expresar el número decimal 47 como una suma de valores de cada dígito.

Solución Como indican sus respectivas posiciones, el dígito 4 tiene un peso de 10, que es 10^1. El dígito 7 tiene un peso de 1, que es 10^0.

$$47 = (4 \times 10^1) + (7 \times 10^0)$$
$$= (4 \times 10) + (7 \times 1) = \mathbf{40 + 7}$$

Problema relacionado* Determinar el valor de cada dígito en el número 939.

* Las respuestas se encuentran al final del capítulo.

EJEMPLO 2.2

Expresar el número decimal 568,23 como suma de los valores de cada dígito.

Solución El dígito 5 de la parte entera tiene un peso de 100, que es 10^2, el dígito 6 tiene un peso de 10, que es 10^1, el dígito 8 tiene un peso de 1, que es 10^0; el dígito 2 de la parte fraccionaria tiene un peso de 0,1, es decir, 10^{-1}, y el dígito 3 de la parte fraccionaria tiene un peso de 0,01, que es 10^{-2}.

$$568{,}23 = (5 \times 10^2) + (6 \times 10^1) + (8 \times 10^0) + (2 \times 10^{-1}) + (3 \times 10^{-2})$$
$$= (5 \times 100) + (6 \times 10) + (8 \times 1) + (2 \times 0{,}1) + (3 \times 0{,}01)$$
$$= \mathbf{500 + 60 + 8 + 0{,}2 + 0{,}03}$$

Problema relacionado Determinar el valor de cada dígito del número 67,924.



CÓMO USAR LA CALCULADORA

Potencias de diez

Ejemplo Hallar el valor de 10^3.

TI-86

Paso 1. [2nd] [LOG] (10^x)

Paso 2. [3]

Paso 3. [ENTER]

Pantalla: 10 ^ 3 → 1000

TI-36X

Paso 1. [1] [0] [y^x]

Paso 2. [3] [=]

Pantalla: 1000

REVISIÓN DE LA SECCIÓN 2.1

Las respuestas se encuentran al final del capítulo.

1. ¿Qué peso tiene el dígito 7 en los siguientes números?
   (a) 1370 **(b)** 6725 **(c)** 7051 **(d)** 58,72
2. Expresar cada uno de los siguientes números decimales como una suma de los productos obtenida mediante la multiplicación de cada dígito por su peso correspondiente:
   (a) 51 **(b)** 137 **(c)** 1492 **(d)** 106,58

2.2 NÚMEROS BINARIOS

El sistema de numeración binario es simplemente otra forma de representar magnitudes. Es menos complicado que el sistema decimal porque sólo emplea dos dígitos. El sistema decimal con sus diez dígitos es un sistema en base diez; el sistema binario con sus dos dígitos es un sistema en base dos. Los dos dígitos binarios (bits) son 1 y 0. La posición de un 1 o un 0 en un número binario indica su peso; o valor dentro del número, del mismo modo que la posición de un dígito decimal determina el valor de ese dígito. Los pesos de un número binario se basan en las potencias de dos.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Contar en binario. ■ Determinar el mayor número decimal que se puede representar con un número dado de bits. ■ Convertir un número binario en un número decimal.

Contar en binario

▲ *El sistema de numeración binario utiliza dos dígitos (bits).*

▲ *El sistema de numeración binario es un sistema en base 2.*

▲ *El valor de un bit se determina por su posición dentro del número.*

Para aprender a contar en el sistema binario, en primer lugar es preciso observar cómo se cuenta en el sistema decimal. Comenzamos en cero y continuamos hasta el nueve antes de quedarnos sin dígitos. Luego, comenzamos con otra posición de dígito (a la izquierda) y continuamos contando desde 10 hasta 99. En este punto, se terminan todas las combinaciones con dos dígitos, por lo que es necesaria una tercera posición de dígito para poder contar desde 100 hasta 999.

Cuando contamos en binario se produce un situación similar, excepto en que sólo disponemos de dos dígitos, denominados *bits*. Empezamos a contar: 0, 1. En este punto, ya hemos utilizado los dos dígitos, por lo que incluimos otra posición de dígito y continuamos: 10, 11. Ahora, hemos agotado todas las combinaciones de dos dígitos, por lo que es necesaria una tercera posición. Con tres posiciones de dígito podemos continuar contando: 100, 101, 110 y 111. Ahora necesitamos una cuarta posición de dígito para continuar, y así sucesivamente. En la Tabla 2.1 se muestra cómo se cuenta desde cero hasta quince. Observe en cada columna la alternancia de 1s y 0s.

Como puede ver en la Tabla 2.1, se necesitan cuatro bits para contar de 0 a 15. En general, con n bits se puede contar hasta un número igual a $2^n - 1$.

$$\text{Máximo número decimal} = 2^n - 1$$

Por ejemplo, con cinco bits ($n = 5$) podemos contar desde cero hasta treinta y uno.

$$2^5 - 1 = 32 - 1 = 31$$

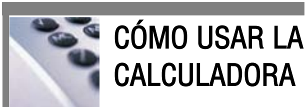
Con seis bits ($n = 6$) podemos contar desde cero hasta sesenta y tres.

$$2^6 - 1 = 64 - 1 = 63$$

En el Apéndice A se proporciona una tabla de las potencias de 2.

Número decimal	Número binario			
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

TABLA 2.1



CÓMO USAR LA CALCULADORA

Potencias de dos

Ejemplo Hallar el valor de 2^5.

TI-86 **Paso 1.** 2 ^

Paso 2. 5 ENTER

2 ^ 5
32

TI-36X **Paso 1.** 2 y^x

Paso 2. 5 =

32

Una aplicación

Aprender a contar en binario le ayudará a entender básicamente cómo pueden utilizarse los circuitos digitales para contar sucesos. Puede tratarse de cualquier cosa, desde elementos que contar en una línea de montaje hasta operaciones de recuento en una computadora. Tomemos un sencillo ejemplo para contar las pelotas de tenis que se desplazan por una cinta transportadora hasta meterse en una caja. Supongamos que en cada caja se introducen nueve pelotas.

El contador mostrado en la Figura 2.1 cuenta los pulsos procedentes de un sensor que detecta el paso de una pelota y genera una secuencia de niveles lógicos (señales digitales) en cada una de sus cuatro salidas para-

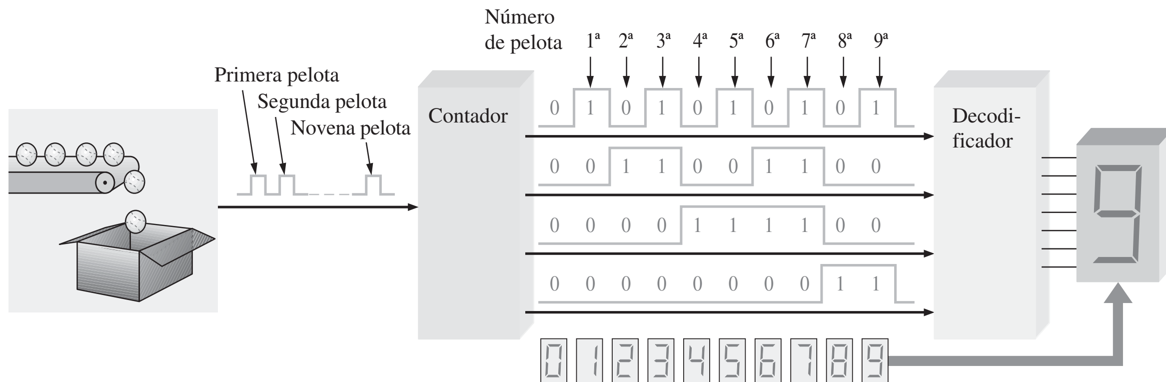


FIGURA 2.1 Ilustración de una sencilla aplicación de recuento binario.

lelas. Cada conjunto de niveles lógicos representa un número binario de 4 bits (ALTO (H) = 1 y BAJO (L) = 0), como se indica. Cuando el decodificador recibe estas señales, decodifica cada conjunto de cuatro bits y lo convierte en el correspondiente número decimal en el display de 7 segmentos. Cuando el contador alcanza el estado binario 1001, quiere decir que ha contado nueve pelotas, el display muestra el número 9 y una nueva caja se desplaza por la cinta transportadora. Entonces el contador se pone a cero (0000) y el proceso comienza de nuevo. El número 9 se ha utilizado en interés de la simplicidad de ofrece un único dígito.

La estructura de pesos de los números binarios

▲ *En un número binario el peso o valor de un bit aumenta de derecha a izquierda.*

Un número binario es un número con peso. El bit más a la derecha es el ***LSB*** (*Least Significant Bit*, bit menos significativo) en un número binario entero y tiene un peso de $2^0 = 1$. El bit más a la izquierda es el ***MSB*** (*Most Significant Bit*, bit más significativo); su peso depende del tamaño del número binario.

Los números fraccionarios también pueden representarse en el sistema binario colocando bits a la derecha de la coma binaria, del mismo modo que los números decimales fraccionarios se colocan a la derecha de la coma decimal. En un número binario con parte fraccionaria, el bit más a la izquierda es el MSB y tiene un peso de $2^{-1} = 0{,}5$. Los pesos fraccionarios de los respectivos bits decrecen de izquierda a derecha según las potencias negativas de dos para cada bit.

La estructura de pesos de un número binario es:

$$2^{n-1} \ldots 2^3\, 2^2\, 2^1\, 2^0 , 2^{-1}\, 2^{-2} \ldots 2^{-n}$$

↑ Coma binaria

donde n es el número de bits a partir de la coma binaria. Por tanto, todos los bits a la izquierda de la coma binaria tienen pesos que son potencias positivas de dos, como previamente se ha visto para los números enteros. Todos los bits situados a la derecha de la coma binaria tienen pesos que son potencias negativas de dos, o pesos fraccionales.



NOTAS INFORMÁTICAS

Las computadoras utilizan los números binarios para seleccionar posiciones de memoria. Cada posición se asigna a un número unívoco denominado *dirección*. Por ejemplo, algunos procesadores Pentium tienen 32 líneas de dirección que pueden seleccionar 2^{32} (4.294.967.296) posiciones diferentes.

Las potencias de dos y sus pesos decimales equivalentes para un número entero binario de 8 bits y un número binario fraccionario de 6 bits se muestran en la Tabla 2.2. Observe que el peso se duplica para cada potencia positiva de dos y que se reduce a la mitad para cada potencia negativa de dos. Puede ampliar fácilmente esta tabla duplicando el peso de la potencia positiva de dos más significativa y dividiendo por dos el peso de la potencia negativa de dos menos significativa; por ejemplo, $2^9 = 512$ y $2^{-7} = 0{,}0078125$.

Potencias positivas de dos (números enteros)									**Potencias negativas de dos (números fraccionarios)**					
$\mathbf{2^8}$	$\mathbf{2^7}$	$\mathbf{2^6}$	$\mathbf{2^5}$	$\mathbf{2^4}$	$\mathbf{2^3}$	$\mathbf{2^2}$	$\mathbf{2^1}$	$\mathbf{2^0}$	$\mathbf{2^{-1}}$	$\mathbf{2^{-2}}$	$\mathbf{2^{-3}}$	$\mathbf{2^{-4}}$	$\mathbf{2^{-5}}$	$\mathbf{2^{-6}}$
256	128	64	32	16	8	4	2	1	1/2	1/4	1/8	1/16	1/32	1/64
									0,5	0,25	0,125	0,0625	0,03125	0,015625

TABLA 2.2 Pesos binarios.

Conversión binario a decimal

▲ *Sume los pesos de todos los 1s de un número binario para obtener el correspondiente valor decimal.*

El valor decimal de cualquier número binario puede hallarse sumando los pesos de todos los bits que están a 1 y descartando los pesos de todos los bits que son 0.

EJEMPLO 2.3

Convertir el número entero binario 1101101 a decimal.

Solución Se determina el peso de cada bit que está a 1, y luego se obtiene la suma de los pesos para obtener el número decimal.

Peso: $2^6\ 2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$

Número binario: 1 1 0 1 1 0 1

$$1101101 = 2^6 + 2^5 + 2^3 + 2^2 + 2^0$$
$$= 64 + 32 + 8 + 4 + 1 = \mathbf{109}$$

Problema relacionado Convertir el número binario 10010001 a decimal.

EJEMPLO 2.4

Convertir el número binario fraccionario 0,1011 a decimal.

Solución Se determina el peso de cada bit que está a 1, y luego se suman los pesos para obtener la fracción decimal.

Peso: $2^{-1}\ 2^{-2}\ 2^{-3}\ 2^{-4}$

Número binario: 0, 1 0 1 1

$$0{,}1011 = 2^{-1} + 2^{-3} + 2^{-4}$$
$$= 0{,}5 + 0{,}125 + 0{,}0625 = \mathbf{0{,}6875}$$

Problema relacionado Convertir el número binario 10,111 a decimal.

REVISIÓN DE LA SECCIÓN 2.2

1. ¿Cuál es el mayor número decimal que puede representarse en binario con ocho bits?
2. Determinar el peso del 1 en el número binario 10000.
3. Convertir el número binario 10111101.011 a decimal.

2.3 CONVERSIÓN DECIMAL A BINARIO

En la Sección 2.2 hemos aprendido a convertir un número binario en su número decimal equivalente. Ahora vamos a estudiar dos formas de convertir un número decimal en un número binario.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Convertir un número decimal en binario utilizando el método de la suma de pesos. ■ Convertir un número decimal entero en un número binario usando el método de la división sucesiva por 2. ■ Convertir a binario un número decimal con parte fraccionaria usando el método de la multiplicación sucesiva por 2.

Método de la suma de pesos

▲ *Para obtener el número binario correspondiente a un número decimal dado, halle los pesos binarios que sumados darán dicho número decimal.*

Una forma de hallar el número binario equivalente a un número decimal determinado consiste en determinar el conjunto de pesos binarios cuya suma es igual al número decimal. Una forma fácil de recordar los pesos binarios es que el peso más bajo es 1, es decir 2^0, y que duplicando cualquier peso, se obtiene el siguiente peso superior; por tanto, la lista de los siete primeros pesos binarios será: 1, 2, 4, 8, 16, 32, 64, como verá en una sección posterior. Por ejemplo, el número decimal 9 puede expresarse como la suma de pesos binarios siguiente:

$$9 = 8 + 1 \quad \text{o} \quad 9 = 2^3 + 2^0$$

Colocando los 1s en las posiciones de pesos apropiadas, 2^3 y 2^0, y los 0s en las posiciones 2^2 y 2^1 se determina el número binario correspondiente al decimal 9.

2^3 2^2 2^1 2^0

1 0 0 1 Número binario para el decimal 9

EJEMPLO 2.5

Convertir a binario los siguientes números decimales:

(a) 12 **(b)** 25 **(c)** 58 **(d)** 82

Solución

(a) $12 = 8 + 4 = 2^3 + 2^2$ ⟶ **1100**

(b) $25 = 16 + 8 + 1 = 2^4 + 2^3 + 2^0$ ⟶ **11001**

(c) $58 = 32 + 16 + 8 + 2 = 2^5 + 2^4 + 2^3 + 2^1$ ⟶ **111010**

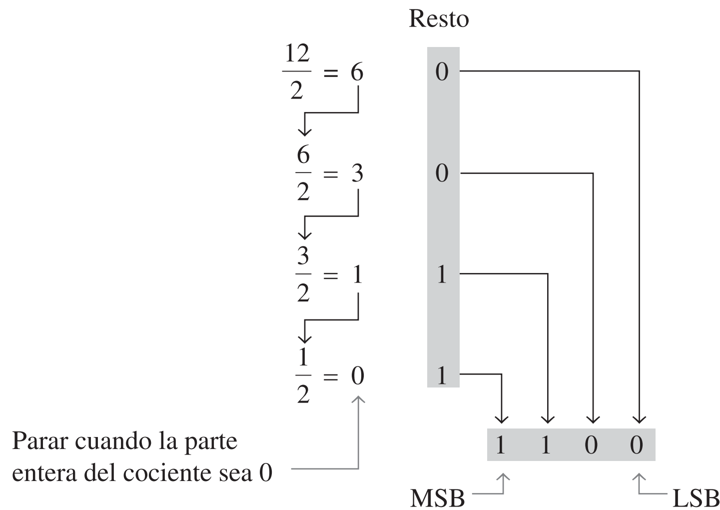
(d) $82 = 64 + 16 + 2 = 2^6 + 2^4 + 2^1$ ⟶ **1010010**

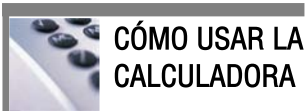
Problema relacionado Convertir a binario el número decimal 125.

Método de la división sucesiva por 2

▲ *Para obtener el número binario correspondiente a un número decimal dado, divida el número decimal entre 2 hasta obtener un cociente igual a 0. Los restos forman el número binario.*

Un método sistemático para convertir a binario números enteros decimales es el proceso de la *división sucesiva por dos.* Por ejemplo, para convertir el número decimal 12 a binario, comenzamos dividiendo 12 entre 2. A continuación, cada cociente resultante se divide entre dos hasta obtener un cociente cuya parte entera sea igual a 0. Los **restos** generados en cada división forman el número binario. El primer resto es el bit menos significativo (LSB) del número binario y el último resto es el bit más significativo (MSB). Este procedimiento se muestra en los pasos siguientes para la conversión a binario del número decimal 12.





CÓMO USAR LA CALCULADORA

Conversión a binario de un número decimal

Ejemplo Convertir a binario el número decimal 57.

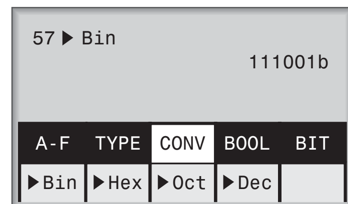
TI-86

Paso 1. 2nd 1 F3 (BASE)

Paso 2. 5 7

Paso 3. F1

Paso 4. ENTER



TI-36X

Paso 1. 3rd EE (DEC)

Paso 2. 5 7

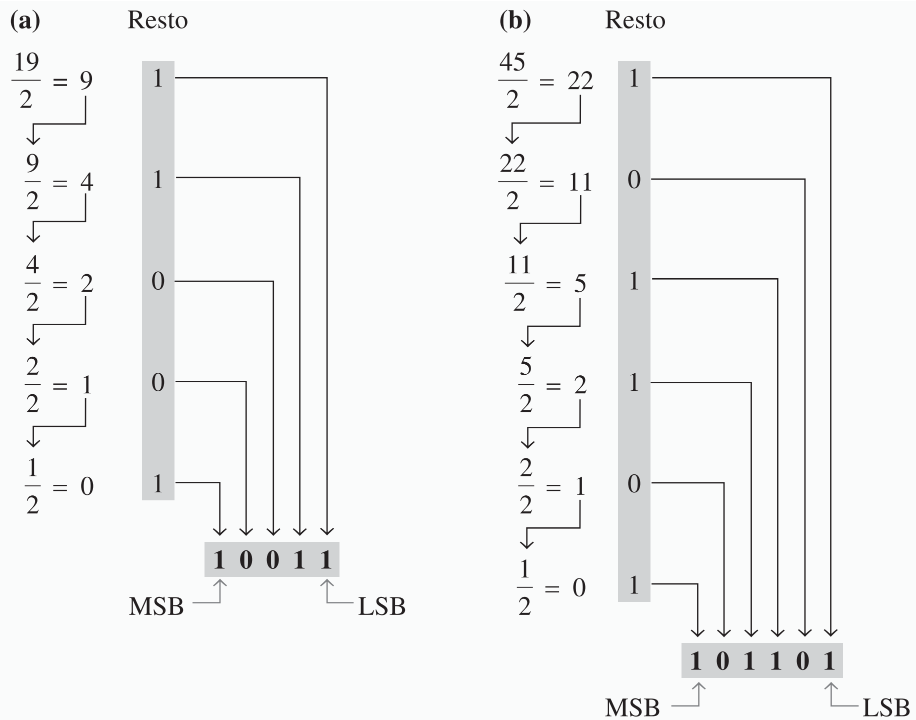
Paso 3. 3rd X (BIN)

111001

EJEMPLO 2.6

Convertir a binario los siguientes números decimales: **(a)** 19 **(b)** 45

Solución



Problema relacionado Convertir a binario el número decimal 39.

Conversión de fracciones decimales a binario

En los Ejemplos 2.5 y 2.6 se han mostrado conversiones de números enteros. Ahora vamos a ver las conversiones de número fraccionarios. Una forma fácil de recordar los pesos binarios fraccionarios es que el peso más significativo es 0,5, es decir 2^{-1}, y que dividiendo entre dos cualquier peso se obtiene el siguiente peso menor; luego una lista de los cuatro primeros pesos binarios fraccionarios sería: 0,5; 0,25; 0,125; 0,0625.

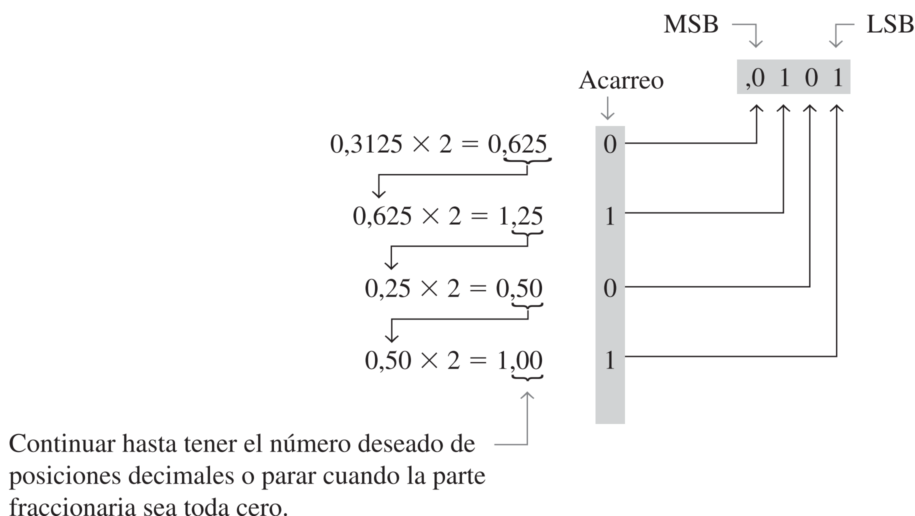
Suma de pesos. El método de la suma de pesos se puede aplicar a los números decimales fraccionarios, como se muestra en el siguiente ejemplo:

$$0{,}625 = 0{,}5 + 0{,}125 = 2^{-1} + 2^{-3} = 0{,}101$$

Lo que indica que hay un 1 en la posición 2^{-1}, un 0 en la posición 2^{-2} y un 1 en la posición 2^{-3}.

Multiplicación sucesiva por 2. Como hemos visto, los números decimales enteros pueden convertirse a binario dividiendo sucesivamente entre dos. Los números decimales fraccionarios pueden convertirse en números binarios multiplicando sucesivamente por 2. Por ejemplo, para convertir a binario el número decimal fraccionario 0,3125, comenzamos multiplicando 0,3125 por 2 y después se multiplica cada parte fraccional resultante del producto por 2 hasta que el producto fraccionario sea cero o hasta que se alcance el número deseado de posiciones decimales. Los dígitos acarreados o, **acarreos**, generados por las multiplicaciones dan lugar al

número binario. El primer acarreo que se obtiene es el MSB y el último acarreo es el LSB. Este procedimiento se ilustra como sigue:



REVISIÓN DE LA SECCIÓN 2.3

1. Convertir a binario cada uno de los números decimales siguientes utilizando el método de la suma de pesos:
   (a) 23 **(b)** 57 **(c)** 45,5
2. Convertir a binario cada uno de los números decimales siguientes utilizando el método de las divisiones sucesivas por 2 (multiplicaciones sucesivas por dos para números fraccionarios):
   (a) 14 **(b)** 21 **(c)** 0,375

2.4 ARITMÉTICA BINARIA

La aritmética binaria es esencial en todas las computadoras digitales y en muchos otros tipos de sistemas digitales. Para entender los sistemas digitales, es necesario conocer los fundamentos de la suma, la resta, la multiplicación y la división binarias. En esta sección se proporciona una introducción que será ampliada en las secciones siguientes.

Después de completar esta sección, el lector deberá ser capaz de:

■ Sumar números binarios. ■ Restar números binarios. ■ Multiplicar números binarios. ■ Dividir números binarios.

Suma binaria

Las cuatro reglas básicas para sumar dígitos binarios son:

0 + 0 = 0	Suma 0 con acarreo 0
0 + 1 = 1	Suma 1 con acarreo 0
1 + 0 = 1	Suma 1 con acarreo 0
1 + 1 = 10	Suma 0 con acarreo 1

▲ *Recuerde, en binario 1+ 1 = 10, no 2.*

Observe que las tres primeras reglas dan lugar a un resultado de un solo bit y la cuarta regla, la suma de dos 1s, da lugar a 2 en binario (10). Cuando se suman números binarios, teniendo en cuenta la última regla se obtiene en la columna dada la suma de 0 y un acarreo de 1 que pasa a la siguiente columna de la izquierda, tal y como se muestra en la siguiente suma de 11 + 1:

Acarreo	Acarreo	
1 ←	1 ←	
0	1	1
+ 0	0	1
1	0	0

En la columna de la derecha 1 + 1 = 0 con acarreo 1, que pasa a la siguiente columna de la izquierda. En la columna central, 1 + 1 + 0 = 0 con acarreo 1, que pasa a la siguiente columna de la izquierda. Y en la columna de la izquierda, 1 + 0 + 0 = 1.

Cuando existe un acarreo igual a 1, se produce una situación en la que se deben sumar tres bits (un bit de cada uno de los números y un bit de acarreo). Esta situación se ilustra como sigue:

Bits de acarreo ↓

1 + 0 + 0 = 01	Suma de 1 con acarreo 0
1 + 1 + 0 = 10	Suma de 0 con acarreo 1
1 + 0 + 1 = 10	Suma de 0 con acarreo 1
1 + 1 + 1 = 11	Suma de 1 con acarreo 1

EJEMPLO 2.7

Sumar los siguientes números binarios:

(a) 11 + 11 **(b)** 100 + 10 **(c)** 111 + 11 **(d)** 110 + 100

Solución La suma decimal equivalente también se muestra como referencia.

(a)	11	3	**(b)**	100	4	**(c)**	111	7	**(d)**	110	6
	+11	+3		+10	+2		+11	+3		+100	+4
	110	6		**110**	6		**1010**	10		**1010**	10

Problema relacionado Sumar 1111 y 1100.

Resta binaria

Las cuatro reglas básicas para la resta de números binarios son:

0 – 0 = 0	
1 – 1 = 0	
1 – 0 = 1	
10 – 1 = 1	0 – 1 con acarreo negativo de 1

▲ *Recuerde, en binario, 10 – 1 = 1, no 9.*

Cuando se restan números, algunas veces se genera un acarreo negativo que pasa a la siguiente columna de la izquierda. En binario, sólo se produce un acarreo negativo cuando se intenta restar 1 de 0. En este caso, cuando se acarrea un 1 a la siguiente columna de la izquierda, en la columna que se está restando se genera

un 10, y entonces debe aplicarse la última de las cuatro reglas enumeradas. Los Ejemplos 2.8 y 2.9 ilustran la resta binaria y se muestra también la resta decimal equivalente.

EJEMPLO 2.8

Realizar las siguientes restas binarias:

(a) 11 − 01 **(b)** 11 − 10

Solución

(a)	11	3	**(b)** 11	3
	−01	−1	−10	−2
	10	2	**01**	1

En este ejemplo no se han generado acarreos negativos. El número binario 01 es el mismo que el 1.

Problema relacionado Restar 100 de 111.

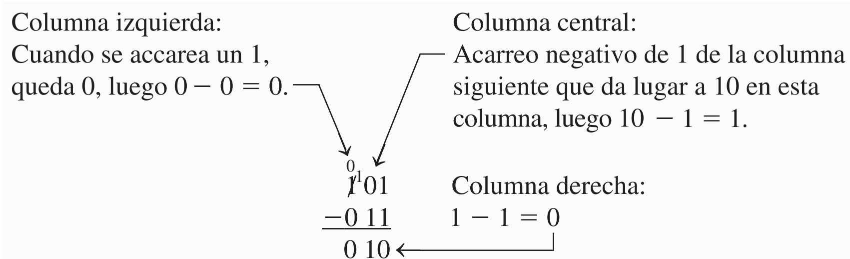
EJEMPLO 2.9

Restar 011 de 101.

Solución

101	5
−011	−3
010	2

Examinemos detalladamente cómo se ha obtenido la resta de los dos números binarios, ya que es necesario un acarreo negativo. Empezamos por la columna de la derecha.



Problema relacionado Restar 101 de 110.

Multiplicación binaria

Las cuatro reglas básicas de la multiplicación de bits son las siguientes:

$0 \times 0 = 0$
$0 \times 1 = 0$
$1 \times 0 = 0$
$1 \times 1 = 1$

▲ *La multiplicación binaria de dos bits es igual que la multiplicación de los dígitos decimales 0 y 1.*

La multiplicación con números binarios se realiza de la misma forma que con números decimales. Se realizan los productos parciales, desplazando cada producto parcial sucesivo una posición hacia la izquierda, y sumando luego todos los productos parciales. El Ejemplo 2.10 ilustra el procedimiento; se muestran las multiplicaciones decimales equivalente por referencia.

EJEMPLO 2.10

Realizar las siguientes multiplicaciones binarias:

(a) 11 × 11 **(b)** 101 × 111

Solución

(a)

```
                 11      3
               × 11    × 3
               ----    ---
Productos  {     11      9
parciales  {   + 11
               ----
               1001
```

(b)

```
                111      7
              × 101     ×5
              -----     --
Productos  {    111     35
parciales  {   000
            + 111
             ------
             100011
```

Resultados: **1001**; **100011**

Problema relacionado Multiplicar 1101 × 1010.

División binaria

▲ *Puede utilizarse una calculadora para realizar operaciones aritméticas con números binarios siempre y cuando no se exceda la capacidad de la calculadora.*

La división binaria sigue el mismo procedimiento que la división decimal, como ilustra el Ejemplo 2.11. También se facilitan las divisiones decimales equivalentes.

EJEMPLO 2.11

Realizar las siguientes divisiones binarias:

(a) 110 ÷ 11 **(b)** 110 ÷ 10

Solución

(a)

```
     10       2
   -----     ---
11)110      3)6
    11        6
   ---       ---
   000        0
```

(b)

```
     11       3
   -----     ---
10)110      2)6
    10        6
   ---       ---
     10       0
     10
     --
     00
```

Cocientes: **10**; **11**

Problema relacionado Dividir 1100 entre 100.

REVISIÓN DE LA SECCIÓN 2.4

1. Realizar las siguientes sumas binarias:
   (a) 1101 + 1010 **(b)** 10111 + 01101
2. Realizar las siguientes restas binarias:
   (a) 1101 − 0100 **(b)** 1001 − 0111
3. Realizar las operaciones binarias indicadas:
   (a) 110 × 111 **(b)** 1100 ÷ 011

2.5 COMPLEMENTO A 1 Y COMPLEMENTO A 2 DE LOS NÚMEROS BINARIOS

El complemento a 1 y el complemento a 2 de un número binario son importantes porque permiten la representación de números negativos. La aritmética en complemento a 2 se usa comúnmente en las computadoras para manipular los números negativos.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Pasar un número binario a su formato en complemento a 1. ■ Pasar un número binario a su formato en complemento a 2 utilizando cualquiera de los dos posibles métodos.

Cálculo del complemento a 1

▲ *Cambie cada uno de los bits del número para obtener el complemento a 1.*

El **complemento** a 1 de un número binario se halla cambiando todos los 1s por 0s y todos los 0s por 1s, como se ilustra a continuación:

1 0 1 1 0 0 1 0 Número binario
↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓
0 1 0 0 1 1 0 1 Complemento a 1

La forma más sencilla de obtener el complemento a 1 de un número binario mediante un circuito digital es utilizando inversores en paralelo (circuitos NOT), como se muestra en la Figura 2.2 para un número binario de 8 bits.

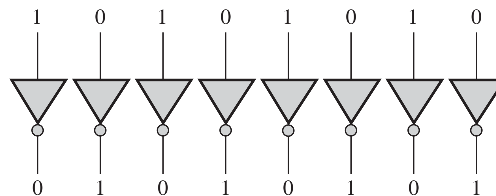


FIGURA 2.2 Ejemplo de inversores utilizados para obtener el complemento a 1 de un número binario.

Cálculo del complemento a 2

▲ *Sume 1 al complemento a 1 para obtener el complemento a 2.*

El complemento a 2 de un número binario se obtiene sumando 1 al bit menos significativo del complemento a 1.

Complemento a 2 = Complemento a 1 + 1

▲ *Cambie todos los bits situados a la izquierda del 1 menos significativo para obtener el complemento a 2.*

Un método alternativo para obtener el complemento a 2 de un número binario es el siguiente:

1. Se empieza por la derecha con el LSB y se escriben los bits como están hasta encontrar el primer 1, incluido éste.
2. Se calcula el complemento a 1 de los bits restantes.

EJEMPLO 2.12

Hallar el complemento a 2 de 10110010:

Solución

10110010	Número binario
01001101	Complemento a 1
+ 1	Sumar 1
01001110	Complemento a 2

Problema relacionado Determinar el complemento a 2 de 11001011.

EJEMPLO 2.13

Hallar el complemento a 2 de 10111000 utilizando el método alternativo.

Solución

10111000	Número binario
01001000	Complemento a 2

Complemento a 1 de los bits originales → 0100

Estos bits no varían. (1000)

Problema relacionado Hallar el complemento a 2 de 11000000.

El complemento a 2 de un número binario negativo puede obtenerse empleando inversores y un sumador, como se indica en la Figura 2.3. Ésta ilustra cómo puede convertirse un número de 8 bits en su complemento a 2, invirtiendo en primer lugar cada bit (obteniendo el complemento a 1) y sumando después 1 al complemento con el sumador.

Para convertir un número en complemento a 1 o en complemento a 2 al formato binario real (no complementado) se usan los dos mismos procedimientos que acabamos de describir. Para convertir el complemento

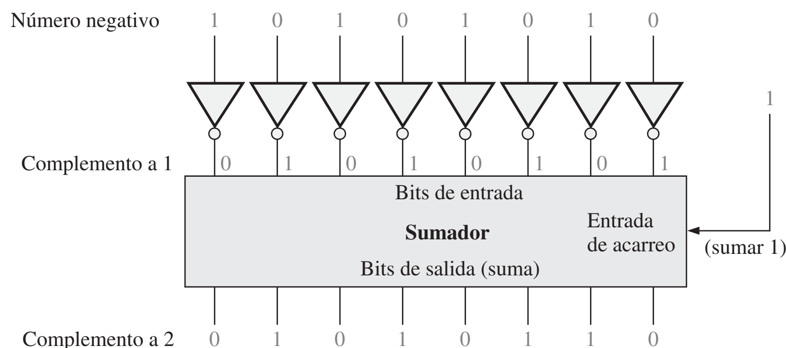


FIGURA 2.3 Ejemplo de obtención de complemento a 2 de un número binario negativo.

a 1 al binario real, se invierten todos los bits. Para convertir el complemento a 2 al binario real, primero se calcula el complemento a 1 y se suma 1 al bit menos significativo.

REVISIÓN DE LA SECCIÓN 2.5

1. Determinar el complemento a 1 de cada uno de los siguientes número binarios:
   (a) 00011010 **(b)** 11110111 **(c)** 10001101
2. Determinar el complemento a 2 de cada uno de los siguientes números binarios:
   (a) 00010110 **(b)** 11111100 **(c)** 10010001

2.6 NÚMEROS CON SIGNO

Los sistemas digitales, como las computadoras, deben ser capaces de manejar números positivos y negativos. Un número binario con signo queda determinado por su magnitud y su signo. El signo indica si se trata de un número positivo o negativo, y la magnitud es el valor del número. Existen tres formatos binarios para representar los número enteros con signo: signo-magnitud, complemento a 1 y complemento a 2. De estos formatos, el complemento a 2 es el más importante y el signo-magnitud es el que menos se emplea. Los números no enteros y muy grandes o muy pequeños pueden expresarse en formato de coma flotante.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Expresar los números positivos y negativos en formato signo-magnitud. ■ Expresar los números positivos y negativos en complemento a 1. ■ Expresar los números positivos y negativos en complemento a 2. ■ Determinar el valor decimal de los números binarios con signo. ■ Expresar un número binario en formato de coma flotante.

Bit de signo

El bit más a la izquierda de un número binario con signo es el **bit de signo**, que indica si el número es positivo o negativo.

Un bit de signo 0 indica que es un número positivo y un bit de signo igual a 1 indica que es un número negativo.

Formato signo-magnitud

Cuando un número binario con signo se representa en formato signo-magnitud, el bit más a la izquierda es el bit de signo y los restantes bits son los bits de magnitud. Los bits de magnitud son el número binario real (no complementado) tanto para los números positivos como para los negativos. Por ejemplo, el número decimal +25 se expresa utilizando un número binario con signo de 8 bits en el formato de signo-magnitud como:

0 0011001

Bit de signo ↑ ↑ Bits de magnitud

El número decimal –25 se expresa como:

10011001

Observe que la única diferencia entre +25 y –25 es el bit de signo, ya que los bits de magnitud representan el número binario real tanto para los números positivos como para los negativos.

En el formato signo-magnitud, un número negativo tiene los mismos bits de magnitud que el correspondiente número positivo, pero el bit de signo es un 1 en lugar de un 0.

Formato del complemento a 1

Los números positivos en el formato de complemento a 1 se representan de la misma forma que los números positivos en el formato signo-magnitud. Sin embargo, los números negativos son el complemento a 1 del correspondiente número positivo. Por ejemplo, con ocho bits, el número decimal –25 se expresa como el complemento a 1 de +25 (00011001), es decir

11100110

En el formato de complemento a 1, un número negativo es el complemento a 1 del correpondiente número positivo.

Formato del complemento a 2

Los números positivos en el formato de complemento a 2 se representan de la misma forma que en el formato signo-magnitud y de complemento a 1. Los números negativos son el complemento a 2 del correspondiente número positivo. De nuevo, utilizando ocho bits, tomamos –25 y lo expresamos como el complemento a 2 de +25 (00011001).

11100111

En el formato de complemento a 2, un número negativo es el complemento a 2 del correspondiente número positivo.



NOTAS INFORMÁTICAS

Las computadoras utilizan el complemento a 2 para realizar operaciones aritméticas con números enteros negativos. La razón de ello es que la sustracción de un número es lo mismo que sumar el complemento a 2 del número. Las computadoras obtienen el complemento a 2 invirtiendo los bits y sumando 1, empleando instrucciones especiales que generan el mismo resultado que el sumador de la Figura 2.3.

EJEMPLO 2.14

Expresar el número decimal –39 como un número de 8 bits en los formatos signo-magnitud, complemento a 1 y complemento a 2.

Solución En primer lugar, escribimos el número de 8 bits para +39.

00100111

En el *formato signo-magnitud,* –39 se obtiene cambiando el bit de signo a 1, y dejando los bits de magnitud como están. El número es:

10100111

En el *formato de complemento a 1,* –39 se obtiene calculando el complemento a 1 de +39 (00100111).

11011000

En el *formato de complemento a 2,* –39 se obtiene calculando el complemento a 1 de +39 (00100111), como sigue

11011000	Complemento a 1
+ 1	
11011001	Complemento a 2

Problema relacionado Expresar –19 y + 19 en los formatos signo-magnitud, complemento a 1 y complemento a 2, con ocho bits.

El valor decimal de los números con signo

Signo-magnitud. Los valores decimales de los números positivos y negativos en el formato signo-magnitud se determinan sumando los pesos de todas las posiciones de los bits de magnitud cuando son 1 e ignorando aquellas posiciones en las que haya ceros. El signo se determina examinando el bit de signo.

EJEMPLO 2.15

Determinar el valor decimal del número binario con signo expresado como signo-magnitud: 10010101.

Solución Los siete bits de magnitud y sus pesos potencias de dos son los siguientes:

2^6	2^5	2^4	2^3	2^2	2^1	2^0
0	0	1	0	1	0	1

Sumando los pesos de las posiciones donde hay 1s, tenemos

$$16 + 4 + 1 = 21$$

El bit de signo es 1; por tanto, el número decimal es **–21.**

Problema relacionado Determinar el valor decimal del número signo-magnitud 01110111.

Complemento a 1. Los valores decimales de los números positivos en el formato de complemento a 1 se determinan sumando los pesos de todas las posiciones de bit donde haya 1 y se ignoran aquellas posiciones donde haya ceros. Los valores decimales de los números negativos se determinan asignando el valor negativo al peso del bit de signo, y sumando todos los pesos donde haya 1s y sumando 1 al resultado.

EJEMPLO 2.16

Determinar los valores decimales de los números binarios con signo expresados en complemento a 1: **(a)** 00010111 **(b)** 11101000

Solución

(a) Los bits y sus pesos según las potencias de dos para el número positivo son:

-2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
0	0	0	1	0	1	1	1

Sumando los pesos donde hay 1s,

$$16 + 4 + 2 + 1 = +\mathbf{23}$$

(b) Los bits y sus pesos según las potencia de dos para el número negativo son los siguientes. Observe que el bit de signo negativo tiene un peso de -2^7, es decir, –128.

-2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	1	1	0	1	0	0	0

Sumando los pesos de las posiciones donde hay 1s,

$$-128 + 64 + 32 + 8 = -24$$

Sumando 1 al resultado, el número decimal final es:

$$-24 + 1 = \mathbf{-23}$$

Problema relacionado Determinar el valor decimal del número en complemento a 1: 11101011.

Complemento a 2. Los valores decimales de los números positivos en el formato de complemento a 2 se determinan sumando los pesos de todas las posiciones de bit donde haya 1 y se ignoran aquellas posiciones donde haya ceros. El peso del bit de signo en un número negativo viene dado por su valor negativo.

EJEMPLO 2.17

Determinar los valores decimales de los siguientes números binarios con signo expresados en complemento a 2 :
(a) 01010110 **(b)** 10101010

Solución **(a)** Los bits y sus pesos según las potencias de dos para el número positivo son:

-2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
0	1	0	1	0	1	1	0

Sumando los pesos donde hay 1s,

$$64 + 16 + 4 + 2 = \mathbf{+86}$$

(b) Los bits y sus pesos según las potencias de dos para el número negativo son los siguientes. Observe que el bit de signo negativo tiene un peso de $-2^7 = -128$.

-2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	0	1	0	1	0	1	0

Sumando los pesos donde hay 1s,

$$-128 + 32 + 8 + 2 = \mathbf{+86}$$

Problema relacionado Determinar el valor decimal del número expresado en complemento a 2: 11010111.

A partir de estos ejemplos, puede deducir una de las razones por las que se prefiere el sistema de complemento a 2 para representar los números con signo: simplemente, es que requiere una adición de pesos independientemente de que el número sea positivo o negativo. El sistema signo-magnitud requiere dos pasos: sumar los pesos de los bits de magnitud y examinar el bit de signo para determinar si el número es positivo o negativo. El sistema de complemento a 1 requiere añadir 1 a la suma de los pesos para los números negativos, pero no para los números positivos. También, generalmente, el sistema de complemento a 1 no se utiliza porque son posibles dos representaciones del cero (00000000 y 11111111).

Rango de representación de los números enteros con signo

▲ El rango de magnitud de un número binario depende del número de bits (n).

Para los ejemplos hemos utilizado números de 8 bits, puesto que la agrupación de 8 bits es un estándar en la mayoría de las computadoras, y recibe el nombre especial de ***byte***. Utilizando un byte u ocho bits, se pueden representar 256 números diferentes. Combinando dos bytes, es decir, dieciséis bits, se pueden representar 65.536 números diferentes. Combinando cuatro bytes, 32 bits, se pueden representar 4,295 $\times 10^9$ números diferentes, y así sucesivamente. La fórmula para calcular el número de combinaciones diferentes de n bits es:

$$\text{Número total de combinaciones} = 2^n$$

Para los números con signo en complemento a 2, el rango de valores para números de n bits es:

$$\text{Rango} = -(2^{n-1}) \text{ hasta } +(2^{n-1} - 1)$$

donde en cada caso hay un bit de signo y $n-1$ bits de magnitud. Por ejemplo, con cuatro bits se pueden representar números en complemento a 2 en el rango de $-(2^3) = -8$ hasta $2^3 - 1 = +7$. Del mismo modo, con ocho bits, se puede abarcar desde −128 hasta +127; con dieciséis bits se puede ir desde −32.768 hasta +32.767, y así sucesivamente.

Números en coma flotante

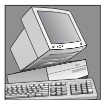
Para representar números **enteros** muy grandes, son necesarios muchos bits. También surge un problema cuando se necesitan representar números con parte entera y parte fraccionaria, tal como 23, 5618. El sistema de numeración en coma flotante, basado en la notación científica, permite representar números muy grandes y números muy pequeños sin aumentar el número de bits, y también sirve para representar números con parte fraccionaria y parte entera.

Un ***número en coma flotante*** (también conocido como *número real*) tiene dos partes más un signo. La **mantisa** es la parte del número en coma flotante que representa la magnitud del número. El **exponente** es la parte de un número en coma flotante que representa el número de lugares que se va a desplazar el punto decimal (o punto binario).

Un ejemplo de número decimal le será útil para comprender el concepto básico de los números en coma flotante. Consideremos un número decimal que, en formato entero, es 241.506.800. La mantisa es .2415068 y el exponente es 9. Cuando el entero se expresa como un número en coma flotante, se normaliza desplazando el punto decimal a la izquierda de todos los dígitos, de modo que la mantisa es un número fraccionario y el exponente es una potencia de 10. Este número en coma flotante se escribe:

$$0{,}2415068 \times 10^9$$

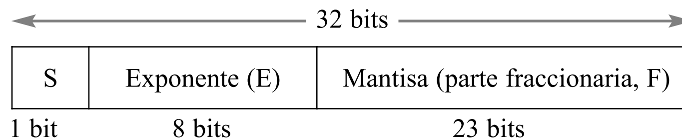
Para los números en coma flotante binarios, el formato definido por el estándar 754–1985 ANSI/IEEE puede tomar tres formas: *simple precisión, doble precisión* y *precisión ampliada*. Todos ellos utilizan el mismo formato básico excepto por el número de bits. Los números en coma flotante de simple precisión tienen 32 bits, los de doble precisión tienen 64 bits y los de precisión ampliada tienen 80 bits. Vamos a restringir nuestra exposición al formato de los números en coma flotante de simple precisión.



NOTAS INFORMÁTICAS

Además de la unidad central de procesamiento (CPU), las computadoras utilizan *coprocesadores* para realizar complejos cálculos matemáticos usando números en coma flotante. El propósito es aumentar el rendimiento liberando a la CPU para que pueda llevar a cabo otras tareas. El coprocesador matemático también se denomina FPU (*Floating-Point Unit*, unidad de coma flotante).

Números binarios en coma flotante de simple precisión. En el formato estándar para un número binario de simple precisión, el bit de signo (S) es el que se encuentra más a la izquierda, el exponente (E) incluye los siguientes 8 bits y la mantisa o parte fraccionaria (F) incluye los restantes 23 bits, como se muestra a continuación.



En la mantisa o parte fraccionaria, se entiende que el punto binario estará a la izquierda de los 23 bits. Realmente, la mantisa consta de 24 bits, ya que, en cualquier número binario, el bit más a la izquierda (más significativo) es siempre un 1. Por tanto, este 1 se entiende que estará allí aunque no ocupe una posición de bit real.

Los 8 bits de los que consta el exponente representan un *exponente desplazado* que se ha obtenido mediante la adición de 127 al exponente real. El propósito de este desplazamiento es poder definir números muy grandes o muy pequeños sin necesidad de emplear un bit de signo diferente para el exponente. El exponente desplazado permite emplear un rango de valores para los exponentes comprendido entre −126 y +128.

Para ilustrar cómo un número binario se expresa en formato de coma flotante, utilizaremos como ejemplo el número binario 1011010010001. En primer lugar, podemos expresarlo como 1 más un número binario fraccionario, desplazando el punto binario doce posiciones a la izquierda y multiplicándolo después por la apropiada potencia de 2.

$$1011010010001 = 1{,}011010010001 \times 2^{12}$$

Suponiendo que se trate de un número positivo, el bit de signo (S) es 0. El exponente, 12, se expresa como un exponente desplazado añadiéndole 127 (12 + 127 = 139). El exponente desplazado (E) se expresa como el número binario 10001011. La mantisa es la parte fraccionaria (F) del número binario, ,011010010001. Dado que siempre existe un 1 a la izquierda de la coma binaria en la expresión de la potencia de dos, no se incluye en la mantisa. El número en coma flotante completo es:

S	E	F
0	10001011	01101001000100000000000

A continuación, veamos cómo evaluar un número binario que ya está en formato de coma flotante. El método general para determinar el valor de un número en coma flotante se expresa mediante la siguiente fórmula:

$$\text{Número} = (-1)^S(1 + F)(2^{E-127})$$

Para ilustrar este método, consideremos el siguiente número binario en coma flotante:

S	E	F
1	10010001	10001110001000000000000

El bit de signo es 1. El exponente desplazado es 10010001 = 145. Aplicando la fórmula, obtenemos:

$$\begin{aligned}\text{Número} &= (-1)^1(1{,}10001110001)(2^{145-127})\\ &= (-1)(1{,}10001110001)(2^{18}) = -1100011100010000000\end{aligned}$$

Este número binario en coma flotante es equivalente a −407.688 en decimal. Dado que el exponente puede ser cualquier número comprendido entre −126 y +128, pueden expresarse números extremadamente grandes y pequeños. Un número en coma flotante de 32 bits puede reemplazar a un número entero binario utilizando 129 bits. Dado que el exponente determina la posición del punto binario, se pueden representar números que contengan tanto parte entera como parte fraccionaria.

Existen dos excepciones para el formato de los números en coma flotante: el número 0,0 se representa utilizando sólo ceros e infinito se representa utilizando sólo unos en el exponente y ceros en la mantisa.

EJEMPLO 2.18

Convertir el número decimal $3{,}248 \times 10^4$ en un número binario en formato de coma flotante de simple precisión.

Solución Convertimos el número decimal a binario.

$$3{,}248 \times 10^4 = 32480 = 111111011100000^2 = 1.11111011100000 \times 2^{14}$$

El bit más significativo (MSB) no ocupará una posición de bit, dado que siempre es 1. Por tanto, la mantisa es el número fraccionario binario de 23 bits 11111011100000000000000 y el exponente desplazado es:

$$14 + 127 = 141 = 10001101_2$$

El número en coma flotante completo es:

0 10001101 11111011100000000000000

Problema relacionado Determinar el valor binario del siguiente número binario en coma flotante:

0 10011000 10000100010100110000000

REVISIÓN DE LA SECCIÓN 2.6

1. Expresar el número decimal +9 como un número binario de 8 bits en formato signo–magnitud.
2. Expresar el número decimal –33 como un número binario de 8 bits en el sistema de complemento a 1.
3. Expresar el número decimal –46 como un número binario de 8 bits en el sistema de complemento a 2.
4. Enumerar las tres partes de un número con signo en formato de coma flotante.

2.7 OPERACIONES ARITMÉTICAS DE NÚMEROS CON SIGNO

En la sección anterior, hemos estudiado cómo se representan, en tres sistemas diferentes, los números con signo. En esta sección, veremos cómo se suman, restan, multiplican y dividen estos números. Puesto que el complemento a 2 es el sistema de representación de números con signo más ampliamente utilizado en las computadoras y en los sistemas basados en microprocesador, esta sección se limita a cubrir la aritmética en complemento a 2. Los procedimientos que se cubren pueden extenderse a los demás sistemas, si fuera necesario.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Sumar números binarios con signo. ■ Explicar cómo las computadoras suman cadenas de números. ■ Definir *desbordamiento* (*overflow*). ■ Restar número binarios con signo. ■ Multiplicar números binarios con signo utilizando el método directo de suma. ■ Multiplicar números binarios con signo utilizando el método de productos parciales. ■ Dividir números binarios con signo.

Suma

Los dos números en una suma se denominan **sumandos**. El resultado es la **suma**. Cuando se suman dos números binarios con signo pueden producirse cuatro casos:

1. Ambos números son positivos.
2. El número positivo es mayor que el negativo en valor absoluto.
3. El número negativo es mayor que el positivo en valor absoluto.
4. Ambos números son negativos.

Veamos caso por caso utilizando números con signo de 8 bits como ejemplos. Como referencia se presentan los números decimales equivalentes.

▲ *La suma de dos números positivos da como resultado un número positivo.*

Ambos números son positivos:

```
  00000111      7
+ 00000100    + 4
  00001011     11
```

La suma es positiva y, por tanto, es un número binario real (no complementado).

▲ *La suma de un número positivo y un número negativo menor en valor absoluto da como resultado un número positivo.*

El número positivo es mayor que el número negativo en valor absoluto:

```
                         00001111     15
                       + 11111010   + −6
Descartar acarreo ——▶ 1  00001001      9
```

El bit de acarreo final no se tiene en cuenta. La suma es positiva y, por tanto, es un número binario real (no complementado).

▲ *La suma de un número positivo y un número negativo mayor en valor absoluto o la suma de dos números negativos da como resultado un número negativo en complemento a 2.*

El número negativo es mayor que el número positivo en valor absoluto:

```
  00010000      16
+ 11101000   + −24
  11111000      −8
```

La suma es negativa y, por tanto, está en complemento a 2.

Ambos números son negativos:

```
                         11111011     −5
                       + 11110111   + −9
Descartar acarreo ——▶ 1  11110010    −14
```

El bit de acarreo final no se tiene en cuenta. Las suma es negativa y, por tanto, está en complemento a 2.

En una computadora, los números negativos se almacenan en formato complemento a 2, por lo que, como puede ver, el procedimiento de suma es muy sencillo: *sumar los dos números y descartar cualquier bit de acarreo final*.

Condición de desbordamiento (overflow). Cuando se suman dos números y el número de bits requerido para representar la suma excede al número de bits de los dos números, se produce un **desbordamiento**, que se indica mediante un bit de signo incorrecto. Un desbordamiento se puede producir sólo cuando ambos números son positivos o negativos. El siguiente ejemplo con números de 8 bits ilustra esta condición.

```
  01111101     125
+ 00111010    + 58
  10110111     183
```

Signo incorrecto (bit 1 de 10110111)
Magnitud incorrecta (0110111)

En este ejemplo, la suma, 183, requiere ocho bits de magnitud. Puesto que los números tienen siete bits de magnitud (un bit es el bit de signo), se produce un acarreo en el bit de signo que da lugar a la indicación de desbordamiento.

***Suma de números de dos en dos*.** Ahora vamos a ver la suma de una cadena de números, sumando de dos en dos. Esto se puede conseguir sumando los dos primeros números, luego se suma el tercer número a la suma de los dos primeros, después se suma el cuarto número al resultado anterior, y así sucesivamente. Así es como las computadoras suman cadenas de números. La suma de números dos a dos se ilustra en el Ejemplo 2.19.

EJEMPLO 2.19

Sumar los números con signo: 01000100, 00011011, 00001110 y 00010010.

Solución Como referencia se proporciona el equivalente decimal.

68	01000100	
+ 27	+ 00011011	Sumar los dos primeros números
95	01011111	Primera suma
+ 14	+ 00001110	Sumar el tercer número
109	01101101	Segunda suma
+ 18	+ 00010010	Sumar el cuarto número
127	**01111111**	Suma final

Problema relacionado Sumar 00110011, 10111111 y 01100011. Son números con signo.

Resta

▲ *La resta es una suma con el signo del sustraendo cambiado.*

La resta es un caso especial de la suma. Por ejemplo, restar +6 (el **sustraendo**) de +9 (el **minuendo**) es equivalente a sumar –6 a +9. Básicamente, *la operación de la resta consiste en cambiar el signo del sustraendo y sumarlo al minuendo. El resultado de una resta se denomina* **diferencia**.

El signo de un número binario positivo o negativo se cambia tomando su complemento a 2.

Por ejemplo, cuando se toma el complemento a 2 del número positivo 00000100 (+4), se obtiene 11111100, que es –4 como demuestra la evaluación de la siguiente suma de pesos:

$$-128 + 64 + 32 + 16 + 8 + 4 =- 4$$

Veamos otro ejemplo, cuando se toma el complemento a 2 del número negativo 11101101 (–19), se obtiene 00010011, que es +19 como demuestra la evaluación de la siguiente suma de pesos:

$$16 + 2 + 1 = 19$$

Puesto que la sustracción o resta es simplemente una suma con el signo del sustraendo cambiado, el proceso se define del siguiente modo:

Para restar dos número con signo, se calcula el complemento a 2 del sustraendo y se suman. Cualquier bit de acarreo final se descarta.

El Ejemplo 2.20 ilustra el proceso de la resta.

EJEMPLO 2.20

Realizar las siguientes restas de números con signo:
(a) 00001000 – 00000011 **(b)** 00001100 – 11110111
(c) 11100111 – 00010011 **(d)** 10001000 – 11100010

Solución Como en otros ejemplos, se facilitan los equivalentes decimales como referencia.

(a) En este caso, 8 – 3 = 8 + (–3) = 5.

	00001000	Minuendo (+8)
	+11111101	Complemento a 2 del sustraendo (–3)
Descartar acarreo →	1 **00000101**	Diferencia (+5)

(b) En este caso, 12 – (–9) = 12 + 9 = 21.

00001100	Minuendo (+12)
+00001001	Complemento a 2 del sustraendo (+9)
00010101	Diferencia (+21)

(c) En este caso, –25 – (+19) =–25 + (–19) = –44.

	11100111	Minuendo (–25)
	+11101101	Complemento a 2 del sustraendo (–19)
Descartar acarreo →	1 **11010100**	Diferencia (–44)

(d) En este caso, –120 – (–30) =–120 + 30 = –90.

10001000	Minuendo (–120)
+00011110	Complemento a 2 del sustraendo (+30)
10100110	Diferencia (–90)

Problema relacionado Restar 01000111 de 01011000.

Multiplicación

▲ *La multiplicación es equivalente a sumar un mismo número el número de veces que indique el multiplicador.*

Los números en una multiplicación se denominan **multiplicando**, **multiplicador** y **producto**. La siguiente multiplicación decimal ilustra estos términos:

8	Multiplicando
× 3	Multiplicador
24	Producto

La operación de la multiplicación en muchas computadoras se realiza utilizando la suma. Como ya ha visto, la sustracción se hace como una suma; ahora vamos a ver cómo se hace la multiplicación.

La *suma directa* y los *productos parciales* son dos métodos básicos para realizar la multiplicación utilizando la suma. En el método de la suma directa, se suma el multiplicando un número de veces igual al multiplicador. En el ejemplo decimal anterior (3 × 8), se suma tres veces el multiplicando: 8 + 8 + 8 = 24. La desventaja de este método es que será muy largo cuando el multiplicador sea un número grande. Si, por ejemplo, multiplica 75 × 350, se debe sumar 75 veces el número 350.

Cuando se multiplican dos números binarios, ambos números deben estar en su formato real (no complementado). El método de suma directa se ilustra en el Ejemplo 2.21 sumando los números binarios de dos en dos.

EJEMPLO 2.21

Multiplicar los números binarios con signo: 01001101 (multiplicando) y 00000100 (multiplicador) utilizando el método de la suma directa.

Solución Puesto que ambos números son positivos, se encuentran en su forma verdadera, y el producto será positivo. El valor decimal del multiplicador es 4, por lo que el multiplicando se suma a sí mismo cuatro veces del siguiente modo:

01001101	Primera vez
+ 01001101	Segunda vez
10011010	Suma parcial
+ 01001101	Tercera vez
11100111	Suma parcial
+ 01001101	Cuarta vez
100110100	Producto

Dado que el bit de signo del multiplicando es 0, no tiene ningún efecto sobre el resultado. Todos los bits del producto son bits de magnitud.

Problema relacionado Multiplicar 01100001 por 00000110 utilizando el método de la suma directa.

El método de los productos parciales es quizá el más común, ya que es la forma de multiplicar manualmente. El multiplicando se multiplica por cada dígito del multiplicador, empezando por el dígito menos significativo. El resultado de la multiplicación del multiplicando por un dígito del multiplicador se denomina *producto parcial*. Cada producto parcial se desplaza una posición a la izquierda y, cuando se han obtenido todos los productos parciales, se suman para obtener el producto final. Aquí tiene un ejemplo con números decimales.

239	Multiplicando
× 123	Multiplicador
717	Primer producto parcial (3 × 239)
478	Segundo producto parcial (2 × 239)
+ 239	Tercer producto parcial (1 × 239)
29.397	Producto final

El signo del producto de una multiplicación depende de los signos del multiplicando y del multiplicador, de acuerdo con las dos reglas siguientes:

- **Si son del mismo signo, el producto es positivo.**
- **Si son de diferente signo, el producto es negativo.**

Los pasos básicos del procedimiento del método de los productos parciales para la multiplicación binaria son los siguientes:

Paso 1. Determinar si los signos del multiplicando y del multiplicador son iguales o diferentes. Así se determina el signo que tendrá el producto.

Paso 2. Poner cualquier número negativo en formato real (no complementado). Puesto que la mayoría de las computadoras almacenan los números negativos en complemento a 2, se requiere la operación de complemento a 2 para obtener el número negativo en formato real.

Paso 3. Empezar por el bit del multiplicador menos significativo y generar los productos parciales. Cuando el bit del multiplicador es 1, el producto parcial es igual al multiplicando. Cuando el bit del multiplicador es 0, el producto parcial es cero. Cada sucesivo producto parcial debe desplazarse un bit a la izquierda.

Paso 4. Sumar cada producto parcial a la suma de los productos parciales anteriores para obtener el producto final.

Paso 5. Si el bit de signo que se había determinado en el paso 1 es negativo, calcular el complemento a 2 del producto. Si es positivo, dejar el producto en formato real. Añadir el bit de signo al producto.

EJEMPLO 2.22

Multiplicar los números binarios con signo: 01010011 (multiplicando) y 11000101 (multiplicador).

Solución

Paso 1. El bit de signo del multiplicando es 0 y el bit de signo del multiplicador es 1. El bit de signo del producto será 1 (negativo).

Paso 2. Calculamos el complemento a 2 del multiplicador para expresarlo en su formato real.

11000101 ⟶ 00111011

Pasos 3 y 4. El proceso de la multiplicación continúa. Observe que en estos pasos sólo se emplean los bits de magnitud.

```
      1010011     Multiplicando
    × 0111011     Multiplicador
      1010011     Primer producto parcial
   + 1010011      Segundo producto parcial
     11111001     Suma del primer y segundo producto
  + 0000000       Tercer producto parcial
    011111001     Suma
 + 1010011        Cuarto producto parcial
   1110010001     Suma
+1010011          Quinto producto parcial
  100011000001    Suma
+ 1010011         Sexto producto parcial
 1001100100001    Suma
+ 0000000         Séptimo producto parcial
 1001100100001    Producto final
```

Paso 5: puesto que el signo del producto es un 1 como se ha determinado en el paso 1, calculamos el complemento a 2 del producto.

1001100100001 ⟶ 0110011011111

Añadir el bit de signo ⟶ **1 0110011011111**

Problema relacionado Verificar que la multiplicación es correcta convirtiendo los números binarios a decimales y realizando la multiplicación.

División

Los números en una división son el **dividendo**, el **divisor** y el **cociente**. Éstos se ilustran en el siguiente formato estándar de división:

$$\frac{\text{dividendo}}{\text{divisor}} = \text{cociente}$$

En las computadoras, la operación de la división se lleva a cabo utilizando la sustracción. Puesto que la sustracción se hace con un sumador, la división también puede realizarse con un sumador.

El resultado de una división es el *cociente*: el cociente es el número de veces que el divisor estará contenido en el dividendo. Esto significa que el divisor puede restarse del dividendo un número de veces igual al cociente, tal como ilustra la división de 21 entre 7:

21	Dividendo
− 7	Primera sustracción del divisor
14	Primer resto parcial
− 7	Segunda sustracción del divisor
7	Segundo resto parcial
− 7	Tercera sustracción del divisor
0	Resto cero

En este sencillo ejemplo, el divisor se ha restado del dividendo tres veces antes de obtener resto cero. Por tanto, el cociente es 3.

El signo del cociente depende de los signos del dividendo y del divisor, de acuerdo con las dos reglas siguientes.

- **Si son del mismo signo, el cociente es positivo.**
- **Si son de diferente signo, el cociente es negativo.**

Cuando se dividen dos números binarios, ambos números deben estar en formato real (no complementado). Los pasos básicos en un procedimiento de división son los siguientes:

Paso 1. Determinar si los signos del dividendo y del divisor son iguales o diferentes. Esto determina qué signo tendrá el cociente. Inicializar el cociente a cero.

Paso 2. Restar el divisor del dividendo utilizando la suma en complemento a 2, para obtener el primer resto parcial, y sumar 1 al cociente. Si este resto parcial es positivo, ir al paso 3. Si el resto parcial es cero o negativo, la división se ha terminado.

Paso 3. Restar el divisor del resto parcial y sumar 1 al cociente. Si el resultado es positivo, repetir para el siguiente resto parcial. Si el resultado es cero o negativo, la división se ha terminado.

Continuar restando el divisor del dividendo y los restos parciales hasta que el resultado sea cero o negativo. Contar el numero de veces que se ha restado el divisor y se obtendrá el cociente. El Ejemplo 2.23 ilustra estos pasos utilizando números binarios con signo de 8 bits.

EJEMPLO 2.23

Dividir 01100100 entre 00011001.

Solución

Paso 1. El signo de ambos números es positivo, por lo que el cociente será positivo. Inicialmente, el cociente es cero: 00000000.

Paso 2. Restar el divisor del dividendo utilizando la suma en complemento a 2 (recuerde que los acarreos finales se descartan).

01100100	Dividendo
+ 11100111	Complemento a 2 del divisor
01001011	Primer resto parcial positivo

Sumar 1 al cociente: 00000000 ± 00000001 = 00000001.

Paso 3: Restar el divisor del primer resto parcial utilizando la suma en complemento a 2.

01001011	Primer resto parcial
+ 11100111	Complemento a 2 del divisor
00110010	Segundo resto parcial positivo

Paso 4: Restar el divisor del segundo resto parcial usando la suma en complemento a 2.

00110010	Segundo resto parcial
+ 11100111	Complemento a 2 del divisor
00011001	Tercer resto parcial positivo

Sumar 1 al cociente: 00000010 + 00000001 = 00000011.

Paso 5: Restar el divisor del tercer resto parcial usando la suma en complemento a 2

00011001	Tercer resto parcial
+ 11100111	Complemento a 2 del divisor
00000000	Resto cero

Sumar 1 al cociente: 00000011 + 00000001 = **00000100** (cociente final). El proceso se ha completado.

Problema relacionado Verificar que el procedimiento es correcto convirtiendo los números binarios a decimales y realizando la división.

REVISIÓN DE LA SECCIÓN 2.7

1. Enumerar los cuatro casos de suma de números.
2. Sumar 00100001 y 10111100.
3. Restar 00110010 de 01110111.
4. ¿Cuál es el signo del producto cuando se multiplican dos números negativos?
5. Multiplicar 01111111 por 00000101.
6. ¿Cuál es el signo del cociente cuando se divide un número positivo entre un número negativo?
7. Dividir 00110000 entre 00001100.

2.8 NÚMEROS HEXADECIMALES

El sistema de numeración hexadecimal consta de dieciséis caracteres y se usan fundamentalmente como una forma simplificada de representar o escribir los números binarios, ya que es muy fácil la conversión entre binario y hexadecimal. Como probablemente habrá comprobado, los números binarios largos son difíciles de leer y escribir, ya que es fácil omitir o transponer un bit. Puesto que las computadoras y microprocesadores sólo entienden los 1s y los 0s, es necesario emplear estos dígitos cuando se programa en “lenguaje máquina”. Imagine tener que escribir una instrucción de sesenta bits para un sistema de microprocesador utilizando 1s y 0s. Es mucho más efectivo utilizar los números hexadecimales u octales. Los números octales se cubren en la Sección 2.9. El sistema hexadecimal se usa frecuentemente en computadoras y aplicaciones de microprocesadores.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Enumerar los caracteres hexadecimales. ■ Contar en hexadecimal. ■ Convertir de binario a hexadecimal. ■ Convertir de hexadecimal a binario. ■ Convertir de hexadecimal a decimal. ■ Convertir de decimal a hexadecimal. ■ Sumar números hexadecimales. ■ Determinar el complemento a 2 de un número hexadecimal. ■ Restar números hexadecimales.

▲ *El sistema de numeración hexadecimal consta de los dígitos 0 a 9 y de las letras A hasta F.*

El sistema **hexadecimal** es un sistema en base dieciséis, es decir, está formado por 16 **caracteres numéricos** y **alfabéticos**. La mayoría de los sistemas digitales procesan grupos de datos binarios que son múltiplos de cuatro bits, lo que hace al número hexadecimal muy adecuado, ya que cada dígito hexadecimal se representa mediante un número binario de 4 bits, como se puede ver en la Tabla 2.3.

Decimal	Binario	Hexadecimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

TABLA 2.3

Diez dígitos numéricos y seis caracteres alfabéticos forman el sistema de numeración hexadecimal. El uso de las letras A, B, C, D, E y F para representar números puede parecer extraño al principio, pero tenga en mente que cualquier sistema de numeración es sólo un conjunto de símbolos secuenciales. Si comprende qué cantidades representan estos símbolos, entonces la forma de los símbolos en sí tiene poca importancia, una vez que se haya acostumbrado a utilizarlos. Utilizaremos el subíndice 16 para designar a los números hexadecimales y evitar así cualquier confusión con los números decimales. En ocasiones, puede ver la letra “h” detrás de un número hexadecimal.



NOTAS INFORMÁTICAS

Con memorias de computadora en el rango de los gigabytes (GB), especificar una dirección de memoria en binario es bastante complicado. Por ejemplo, hay que emplear 32 bits para especificar una dirección de una memoria de 4 GB. Es mucho más sencillo escribir un código de 32 bits usando ocho dígitos hexadecimales.

Contar en hexadecimal

¿Cómo se continúa contando en hexadecimal cuando se ha llegado a la letra F? Simplemente se inicia otra columna y se continúa contando así:

10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 1A, 1B, 1C, 1D, 1E, 1F, 20, 21, 22, 23, 24,
25, 26, 27, 28, 29, 2A, 2B, 2C, 2D, 2E, 2F, 30, 31, . . .

Con dos dígitos hexadecimales, se puede contar hasta FF_{16}, que corresponde al decimal 255. Para continuar contando, se necesitan tres dígitos hexadecimales. Por ejemplo, 100_{16} es el decimal 256, 101_{16} es el deci-

mal 257, y así sucesivamente. El número hexadecimal máximo con 3 dígitos es FFF_{16}, es decir el decimal 4.095. El máximo número hexadecimal con 4 dígitos es el $FFFF_{16}$, que es el decimal 65.535.

Conversión binario-hexadecimal

La conversión de un número binario en hexadecimal es un procedimiento muy sencillo. Simplemente se parte el número binario en grupos de 4 bits, comenzando por el bit más a la derecha, y se reemplaza cada grupo de 4 bits por su símbolo hexadecimal equivalente.

EJEMPLO 2.24

Convertir a hexadecimal los siguientes números binarios:

(a) 1100101001010111 **(b)** 111111000101101001

Solución

(a) 1100 1010 0101 0111 → C A 5 7 = $\mathbf{CA57_{16}}$

(b) 0011 1111 0001 0110 1001 → 3 F 1 6 9 = $\mathbf{3F169_{16}}$

En el apartado (b) se han añadido dos ceros para completar el grupo de 4 bits de la izquierda.

Problema relacionado Convertir el número binario 1001111011110011100 a hexadecimal.

Conversión hexadecimal-binario

▲ *La conversión entre hexadecimal y binario es directa y muy fácil.*

Para convertir un número hexadecimal en un número binario se realiza el proceso inverso, reemplazando cada símbolo hexadecimal por el grupo de cuatro bits adecuado.

Debería estar claro que es mucho más fácil tratar con un número hexadecimal que con el número binario equivalente. Puesto que la conversión también es fácil, el sistema hexadecimal se usa ampliamente para representar los números binarios en programación, salidas de impresora y displays.

EJEMPLO 2.25

Determinar los números binarios correspondientes a los siguientes números hexadecimales:

(a) $10A4_{16}$ **(b)** $CF8E_{16}$ **(c)** 9742_{16}

Solución

(a) 1 0 A 4 → **1 0000 1010 0100**

(b) C F 8 E → **1100 1111 1000 1110**

(c) 9 7 4 2 → **1001 0111 0100 0010**

En el apartado (a), el MSB se entiende que tiene tres ceros delante del 1 para formar un grupo de 4 bits.

Problema relacionado Convertir el número hexadecimal 6BD3 a binario.

Conversión hexadecimal-decimal

Un método para encontrar el equivalente decimal de un número hexadecimal es, primero, convertir el número hexadecimal a binario, y después, el binario a decimal.

EJEMPLO 2.26

Convertir los siguientes números hexadecimales a decimal:

(a) $1C_{16}$ **(b)** $A85_{16}$

Solución Recuerde que primero se hace la conversión del número hexadecimal a binario y luego a decimal.

(a)

$$\overbrace{0001}^{1}\overbrace{1100}^{C} = 2^4 + 2^3 + 2^2 = 16 + 8 + 4 = \mathbf{28}_{10}$$

(b)

$$\overbrace{1010}^{A}\overbrace{1000}^{8}\overbrace{0101}^{5} = 2^{11} + 2^9 + 2^7 + 2^2 + 2^0 = 2048 + 512 + 128 + 4 + 1 = \mathbf{2693}_{10}$$

Problema relacionado Convertir a decimal el número hexadecimal 6BD.

Otro método para convertir un número hexadecimal a su equivalente decimal es multiplicar el valor decimal de cada dígito hexadecimal por su peso, y luego realizar la suma de estos productos. Los pesos de un número hexadecimal crecen según las potencias de 16 (de derecha a izquierda). Para un número hexadecimal de 4 dígitos, los pesos son:

16^3	16^2	16^1	16^0
4096	256	16	1

EJEMPLO 2.27

Convertir los siguientes números hexadecimales a decimal:

(a) $E5_{16}$ **(b)** $B2F8_{16}$

Solución En la Tabla 2.3 puede ver que las letras A hasta F representan los números decimales 10 hasta 15, respectivamente.

(a) $E5_{16} = (E \times 16) + (5 \times 1) = (14 \times 16) + (5 \times 1) + 224 + 5$

$= \mathbf{229}_{10}$

(b) $B2F8_{16} = (B \times 4096) + (2 \times 256) + (F \times 16) + (8 \times 1)$

$= (11 \times 4096) + (2 \times 256) + (15 \times 16) + (8 \times 1)$

$= 45.056 + 512 + 240 + 8$

$= \mathbf{45.816}_{10}$

Problema relacionado Convertir $60A_{16}$ a decimal.



CÓMO USAR LA CALCULADORA

Potencias de 16

Ejemplo Hallar el valor de 16^4.

TI-86 **Paso 1.** 1 6 ^

Paso 2. 4 ENTER

16 ^ 4
65536

TI-36X **Paso 1.** 1 6 y^x

Paso 2. 4 =

65536

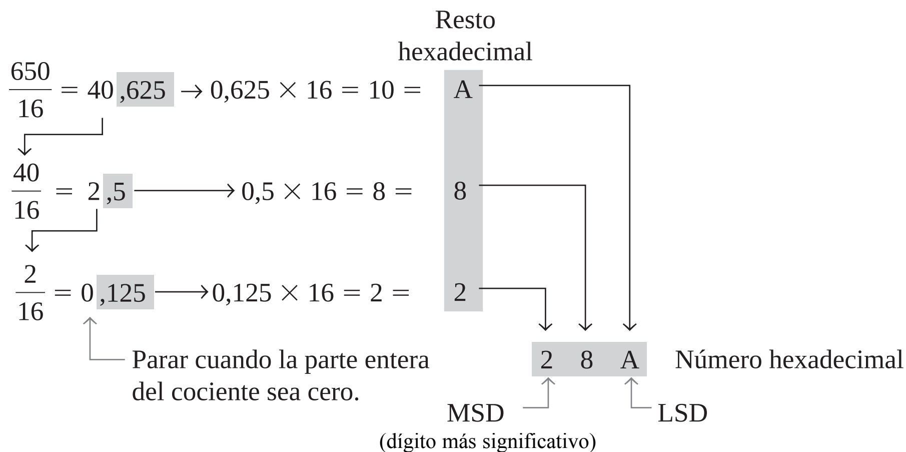
Conversión decimal-hexadecimal

La división sucesiva por 16 de un número decimal generará el número hexadecimal equivalente formado por los restos de las divisiones. El primer resto que se genera es el dígito menos significativo (LSD). Cada división sucesiva por 16 dará un resto que será un dígito del número hexadecimal equivalente. Este procedimiento es similar a la división sucesiva por 2 para la conversión decimal–binario, que se ha visto en la Sección 2.3. El Ejemplo 2.28 ilustra el procedimiento. Observe que cuando un cociente tiene parte fraccionaria, ésta se multiplica por el divisor para obtener el resto.

EJEMPLO 2.28

Convertir el número decimal 650 en hexadecimal mediante el método del división sucesiva por 16.

Solución



Problema relacionado Convertir a hexadecimal el número decimal 2591.

Suma hexadecimal

La suma puede hacerse directamente con números hexadecimales, teniendo en cuenta que los dígitos hexadecimales de 0 a 9 son equivalentes a los dígitos decimales de 0 a 9 y que los dígitos hexadecimales de A

hasta F son equivalentes a los números decimales 10 hasta 15. Cuando se suman dos números hexadecimales se usan las reglas siguientes. (Los números decimales se indican con el subíndice 10.)

▲ *Se puede emplear una calculadora para realizar operaciones aritméticas con números hexadecimales*

1. En cualquier columna dada de una suma, pensar en los dos dígitos hexadecimales en términos de sus valores decimales. Por ejemplo, $5_{16} = 5_{10}$ y $C_{16} = 12_{10}$.
2. Si la suma de los dos dígitos es 15_{10} o menor, reducir al dígito hexadecimal correspondiente.
3. Si la suma de los dos dígitos es mayor que 15_{10}, hay que reducir la suma que excede de 16_{10} y pasar el acarreo de 1 a la siguiente columna.



CÓMO USAR LA CALCULADORA

Conversión de un número decimal en un número hexadecimal

Ejemplo Convertir a hexadecimal el número decimal 650.

TI-86

Paso 1. [2nd] [1] [F3] (BASE)
Paso 2. [6] [5] [0]
Paso 3. [F2]
Paso 4. [ENTER]

650 ▶ Hex
28Ah

A-F TYPE CONV BOOL BIT
▶Bin ▶Hex ▶Oct ▶Dec

TI-36X

Paso 1. [3rd] [EE] (DEC)
Paso 2. [6] [5] [0]
Paso 3. [3rd] [(] (HEX)

28A

EJEMPLO 2.29

Sumar los siguientes números hexadecimales:

(a) $23_{16} + 16_{16}$ **(b)** $58_{16} + 22_{16}$ **(c)** $2B_{16} + 84_{16}$ **(d)** $DF_{16} + AC_{16}$

Solución

(a)
$$\begin{array}{r} 23_{16} \\ +16_{16} \\ \hline \mathbf{39}_{16} \end{array}$$
columna derecha: $3_{16} + 6_{16} = 3_{10} + 6_{10} = 9_{10} = 9_{16}$
columna izquierda: $2_{16} + 1_{16} = 2_{10} + 1_{10} = 3_{10} = 3_{16}$

(b)
$$\begin{array}{r} 58_{16} \\ +\ 22_{16} \\ \hline \mathbf{7A}_{16} \end{array}$$
columna derecha: $8_{16} + 2_{16} = 8_{10} + 2_{10} = 10_{10} = A_{16}$
columna izquierda: $5_{16} + 2_{16} = 5_{10} + 2^{10} = 7_{10} = 7_{16}$

(c)
$$\begin{array}{r} 2B_{16} \\ +\ 84_{16} \\ \hline \mathbf{AF}_{16} \end{array}$$
columna derecha: $B_{16} + 4_{16} = 11_{10} + 4_{10} = 15_{10} = F_{16}$
columna izquierda: $2_{16} + 8_{16} = 2_{10} + 8_{10} = 10_{10} = A_{16}$

(d) DF_{16} + AC_{16} = $\mathbf{18B}_{16}$

columna derecha: $F_{16} + C_{16} = 15_{10} + 12_{10} = 27_{10}$
$27_{10} - 16_{10} = 11_{10} = B_{16}$ con un acarreo de 1

columna izquierda: $D_{16} + A_{16} + 1_{16} = 13_{10} + 10_{10} + 1_{10} = 24_{10}$
$24_{10} - 16_{10} = 8_{10} = 8_{16}$ con un acarreo de 1

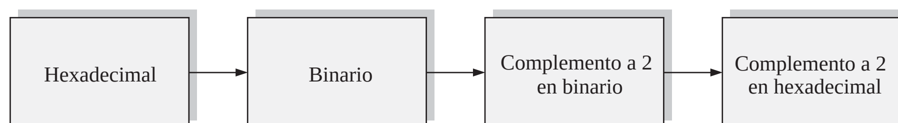
Problema relacionado Sumar $4C_{16}$ y $3A_{16}$

Resta hexadecimal

Como ya hemos visto, el complemento a 2 permite realizar restas sumando números binarios. Puesto que un número hexadecimal se puede usar para representar un número binario, también se puede emplear para representar el complemento a 2 del número binario.

Existen tres formas de obtener el complemento a 2 de un número hexadecimal. El método 1 es el más común y el más fácil de utilizar. Los métodos 2 y 3 son formas alternativas.

Método 1. Se convierte el número hexadecimal a binario. Se calcula el complemento a 2 del número binario. Se convierte el resultado a hexadecimal. Esto se ilustra en la Figura 2.4.



Ejemplo:

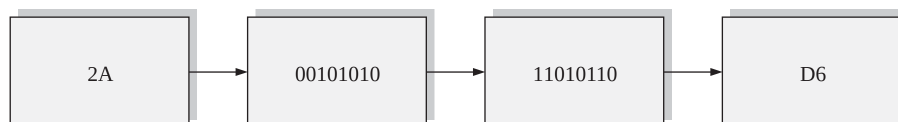
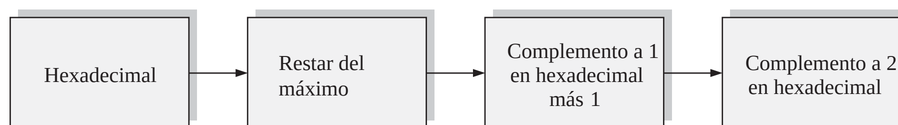


FIGURA 2.4 Obtención del complemento a 2 de un número hexadecimal, método 1.

Método 2. Restar el número hexadecimal del número hexadecimal máximo y sumar 1. Esto se ilustra en la Figura 2.5.



Ejemplo:

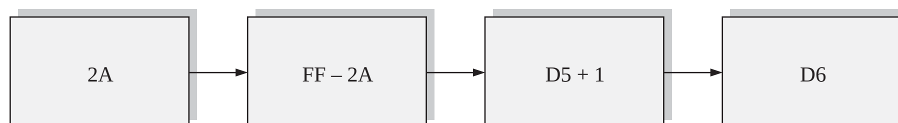


FIGURA 2.5 Obtención del complemento a 2 de un número hexadecimal, método 2.

Método 3. Se escribe la secuencia de dígitos hexadecimales. Se escribe la secuencia inversa debajo de la secuencia directa. El complemento a 1 de cada uno de los dígitos hexadecimales es directamente el dígito que se encuentra debajo. Se suma 1 al número resultante para obtener el complemento a 2. Esto se ilustra en la Figura 2.6.

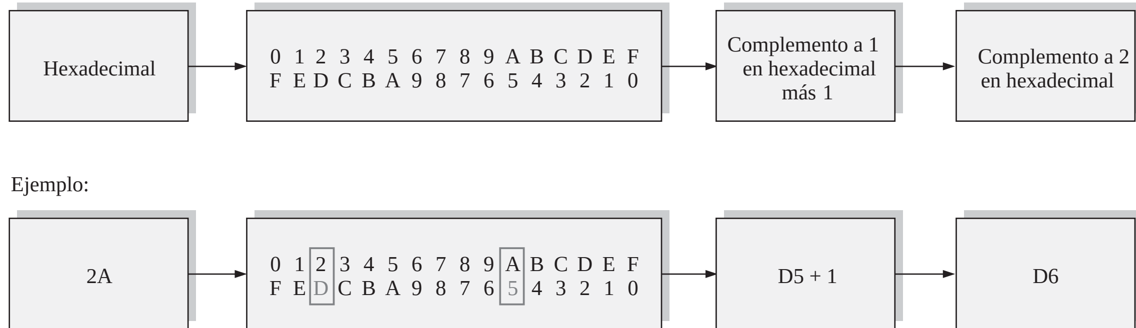


FIGURA 2.6 Obtención del complemento a 2 de un número hexadecimal, método 3.

EJEMPLO 2.30

Restar los siguientes números hexadecimales:

(a) $84_{16} - 2A_{16}$ **(b)** $C3_{16} - 0B_{16}$

Solución **(a)** $2A_{16} = 00101010$

Complemento a 2 de $2A_{16} = 11010110 = D6_{16}$ (usando el método 1)

$$\begin{array}{r} 84_{16} \\ +\ D6_{16} \\ \hline \not{1}5A_{16} \end{array}$$

Suma

el acarreo no se tiene en cuenta, como en la suma en complemento a 2

La diferencia es $\mathbf{5A_{16}}$.

(b) $0B_{16} = 00001011$

Complemento a 2 de $0B_{16} = 11110101 = F5_{16}$ (usando el método 1)

$$\begin{array}{r} C3_{16} \\ +\ F5_{16} \\ \hline \not{1}B8_{16} \end{array}$$

Suma

el acarreo no se tiene en cuenta

La diferencia es $\mathbf{B8_{16}}$.

Problema relacionado Restar 173_{16} de BCD_{16}.

REVISIÓN DE LA SECCIÓN 2.8

1. Convertir a hexadecimal los siguientes números binarios:
   (a) 10110011 **(b)** 110011101000
2. Convertir a binario los siguientes números hexadecimales:
   (a) 57_{16} **(b)** $3A5_{16}$ **(c)** $F80B_{16}$
3. Convertir a decimal el número $9B30_{16}$.

4. Convertir a hexadecimal el número decimal 573.
5. Sumar directamente los siguientes números hexadecimales:
   (a) $18_{16} + 34_{16}$ **(b)** $3F_{16} + 2A_{16}$
6. Restar los siguientes número hexadecimales:
   (a) $75_{16} - 21_{16}$ **(b)** $94_{16} - 5C_{16}$

2.9 NÚMEROS OCTALES

Como el sistema hexadecimal, el sistema octal proporciona un método adecuado para expresar los códigos y números binarios. Sin embargo, se usa menos frecuentemente que el hexadecimal en las computadoras y microprocesadores para expresar magnitudes binarias con propósitos de entrada y salida.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Escribir los dígitos del sistema de numeración octal. ■ Convertir de octal a decimal. ■ Convertir de decimal a octal. ■ Convertir de octal a binario. ■ Convertir de binario a octal.

El sistema de numeración ***octal*** está formado por ocho dígitos, que son:

0, 1, 2, 3, 4, 5, 6, 7

Para contar por encima de 7, añadimos otra columna y continuamos así:

10, 11, 12, 13, 14, 15, 16, 17, 20, 21

▲ *El sistema de numeración octal es un sistema en base 8.*

Contar en octal es parecido a contar en decimal, excepto que los dígitos 8 y 9 no se usan. Para distinguir los números octales de los números decimales y hexadecimales, utilizaremos el subíndice 8 para indicar un número octal. Por ejemplo, 15_8 es equivalente a 13_{10} en decimal y a D en hexadecimal. En ocasiones, puede ver una "o" o una "Q" detrás de un número octal.

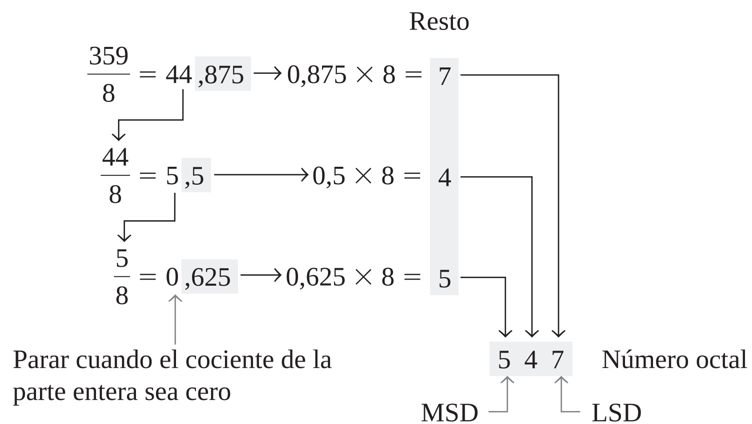
Conversión octal-decimal

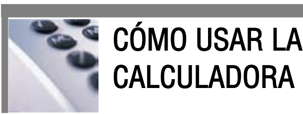
Puesto que el sistema de numeración octal es un sistema en base ocho, cada posición sucesiva de dígito es una potencia superior de ocho, empezando por el dígito situado más a la derecha con 8^0. La evaluación de un número octal en términos de su equivalente decimal se consigue multiplicando cada dígito por su peso y sumando los productos, como se muestra a continuación para 2374_8:

$$\begin{aligned} \text{Peso: } & 8^3\ 8^2\ 8^1\ 8^0 \\ \text{Número octal: } & 2\ \ 3\ \ 7\ \ 4 \\ 2374_8 &= (2 \times 8^3) + (3 \times 8^2) + (7 \times 8^1) + (4 \times 8^0) \\ &= (2 \times 512) + (3 \times 64) + (7 \times 8) + (4 \times 1) \\ &= 1024 + 192 + 56 + 4 = 1276_{10} \end{aligned}$$

Conversión decimal-octal

Un método para convertir un número decimal en un número octal es el método de la división sucesiva por 8, que es parecido al método utilizado en la conversión a binario o a hexadecimal de los números decimales. Para mostrar cómo se hace, convertimos a octal el número decimal 359. Cada división sucesiva por 8 da un resto que será un dígito del número octal equivalente. El primer resto que se genera es el dígito menos significativo (LSD).





CÓMO USAR LA CALCULADORA

Conversión de un número decimal en un número octal

Ejemplo Convertir a octal el número decimal 439.

TI-86

Paso 1. 2nd 1 F3 (BASE)

Paso 2. 4 3 9

Paso 3. F3

Paso 4. ENTER

439 ▶ Oct
667o

A-F TYPE CONV BOOL BIT
▶Bin ▶Hex ▶Oct ▶Dec

TI-36X

Paso 1. 3rd EE (DEC)

Paso 2. 4 3 9

Paso 3. 3rd) (OCT)

667

Conversión octal-binario

▲ *El sistema octal es una forma conveniente de representar los números binarios, aunque no es tan comúnmente utilizado como el hexadecimal.*

Puesto que cada dígito octal se puede representar mediante un número binario de 3 dígitos, es fácil convertir a binario un número octal. Cada dígito octal se representa mediante tres bits, como se muestra en la Tabla 2.4.

Para convertir a binario un número octal basta con reemplazar cada dígito octal con los tres bits apropiados.

Dígito octal	0	1	2	3	4	5	6	7
Binario	000	001	010	011	100	101	110	111

TABLA 2.4 Conversión octal/binario

EJEMPLO 2.31

Convertir a binario cada uno de los siguientes números octales:

(a) 13_8 **(b)** 25_8 **(c)** 140_8 **(d)** 7526_8

Solución

(a) 1 3 → **011011**

(b) 2 5 → **010101**

(c) 1 4 0 → **001100000**

(d) 7 5 2 6 → **111101010110**

Problema relacionado Convertir a decimal cada uno de los números binarios y verificar que los valores concuerdan con el valor decimal del número octal correspondiente.

Conversión binario-octal

La conversión de un número binario a un número octal es el inverso de la conversión de octal a binario. El procedimiento es el siguiente: se comienza por el grupo de tres bits más a la derecha y, moviéndose de derecha a izquierda, se convierte cada grupo de 3 bits en el dígito octal equivalente. Si para el grupo más a la izquierda no hay disponibles tres bits, se añaden uno o dos ceros para completar el grupo. Estos ceros no afectan al valor del número binario.

EJEMPLO 2.32

Convertir a octal cada uno de los siguientes números binarios:

(a) 110101 **(b)** 101111001 **(c)** 100110011010 **(d)** 11010000100

Solución

(a) 110 101 → 6 5 = $\mathbf{65}_8$

(b) 101 111 001 → 5 7 1 = $\mathbf{571}_8$

(c) 100 110 011 010 → 4 6 3 2 = $\mathbf{4632}_8$

(d) 011 010 000 100 → 3 2 0 4 = $\mathbf{3204}_8$

Problema relacionado Convertir a octal el número binario 1010101000111110010.

REVISIÓN DE LA SECCIÓN 2.9

1. Convertir a decimal los siguientes números octales:
   (a) 73_8 **(b)** 125_8
2. Convertir a octal los siguientes números decimales:
   (a) 98_{10} **(b)** 163_{10}
3. Convertir a binario los siguientes números octales:
   (a) 46_8 **(b)** 723_8 **(c)** 5624_8
4. Convertir a octal los siguientes números binarios:
   (a) 110101111
   (b) 1001100010
   (c) 10111111001

2.10 CÓDIGO DECIMAL BINARIO (BCD)

El código decimal binario (BCD, *Binary Coded Decimal*) es una forma de expresar cada uno de los dígitos decimales con un código binario. Puesto que en el sistema BCD sólo existen diez grupos de código, es muy fácil convertir entre decimal y BCD. Como nosotros leemos y escribimos en decimal, el código BCD proporciona una excelente interfaz para los sistemas binarios. Ejemplos de estas interfaces son las entradas por teclado y las salidas digitales.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Convertir cada dígito decimal a BCD. ■ Expresar en BCD números decimales. ■ Convertir de BCD a decimal. ■ Sumar números en BCD.

El código 8421

El código 8421 es un tipo de **código decimal binario (*BCD*)**. Código decimal binario significa que cada dígito decimal, de 0 hasta 9, se representa mediante un código binario de cuatro bits. La designación 8421 indica los pesos binarios de los cuatro bits (2^3, 2^2, 2^1, 2^0). La facilidad de conversión entre los números en código 8421 y los familiares números decimales es la principal ventaja de este código. Todo lo que tiene que recordar sobre las diez combinaciones binarias que representan los diez dígitos decimales se muestra en la Tabla 2.5. El código 8421 es el código BCD más importante, y cuando hacemos referencia a BCD, siempre es al código 8421, a no ser que se indique otra cosa.

Dígito decimal	0	1	2	3	4	5	6	7	8	9
Binario	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

TABLA 2.5 Conversión decimal/BCD.

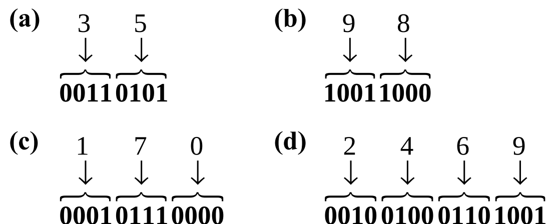
Códigos no válidos. Debería saber que, con cuatro dígitos, se pueden representar dieciséis números (desde 0000 hasta 1111), pero en el código 8421, sólo se usan diez de ellos. Las seis combinaciones que no se emplean (1010, 1011, 1100, 1101, 1110 y 1111) no son válidas en el código BCD 8421.

Para expresar cualquier número decimal en BCD, simplemente reemplace cada dígito decimal por el apropiado código de 4 bits, tal como muestra el Ejemplo 2.33.

EJEMPLO 2.33

Convertir a BCD los siguientes números decimales: **(a)** 35 **(b)** 98 **(c)** 170 **(d)** 2469

Solución



Problema relacionado Convertir a BCD el número decimal 9673.

Es igualmente sencillo determinar el número decimal a partir del código BCD. Se comienza por el bit más a la derecha y se divide el código en grupos de cuatro bits. Después se escribe el dígito decimal representado por cada grupo de 4 bits.

EJEMPLO 2.34

Convertir a decimal cada uno los siguientes códigos BCD:

(a) 10000110 **(b)** 001101010001 **(c)** 1001010001110000

Solución

(a) 1000 0110 → **8** **6**

(b) 0011 0101 0001 → **3** **5** **1**

(c) 1001 0100 0111 0000 → **9** **4** **7** **0**

Problema relacionado Convertir a decimal el código BCD 10000010001001110110.

Suma en BCD

BCD es un código numérico y puede utilizarse en operaciones aritméticas. La suma es la más importante de estas operaciones, ya que las otras tres operaciones (sustracción, multiplicación y división) se pueden llevar a cabo utilizando la suma. A continuación, vamos a ver cómo se suman dos números BCD:

Paso 1. Sumar los dos números BCD utilizando las reglas de la suma binaria vistas en la Sección 2.4.

Paso 2. Si una suma de 4 bits es igual o menor que 9, es un número BCD válido.

Paso 3. Si una suma de 4 bits es mayor que 9, o si genera un acarreo en el grupo de 4 bits, el resultado no es válido. En este caso, se suma 6 (0110) al grupo de 4 bits para saltar así los seis estados no válidos y pasar al código 8421. Si se genera un acarreo al sumar 6, éste se suma al grupo de 4 bits siguiente.

El Ejemplo 2.35 ilustra la suma en BCD para los casos en que la suma en cada columna de 4 bits es igual o menor que 9 y, por tanto, las sumas de 4 bits son números BCD válidos. El Ejemplo 2.36 ilustra el procedimiento en el caso de que se produzcan sumas no válidas (mayores que 9 o que generen acarreo).

EJEMPLO 2.35

Sumar los siguientes números BCD:

(a) 0011 + 0100 **(b)** 00100011 + 00010101

(c) 10000110 + 00010011 **(d)** 010001010000 + 010000010111

Solución Se muestra la suma decimal con propósitos de comparación.

(a)

0011	3
+ 0100	+ 4
0111	7

(b)

0010 0011	23
+ 0001 0101	+15
0011 1000	38

(c)

1000 0110	86
+ 0001 0011	+ 13
1001 1001	99

(d)

0100 0101 0000	450
+ 0100 0001 0111	+ 417
1000 0110 0111	867

Observe que en ningún caso la suma de las columnas de 4 bits excede 9, por lo que los resultados son números BCD válidos.

Problema relacionado Sumar los números BCD: 1001000001000011 + 0000100100100101.

EJEMPLO 2.36

Sumar los siguientes números BCD

(a) 1001 + 0100 **(b)** 1001 + 1001

(c) 00010110 + 00010101 **(d)** 01100111 + 01010011

Solución Se muestra la suma decimal con propósitos de comparación.

(a)

	1001		9
	+0100		+4
	1101	Número BCD no válido (>9)	13
	+0110	Sumar 6	
0001	**0011**	Número BCD válido	
↓	↓		
1	3		

(b)

	1001		9
	+1001		+ 9
1	0010	No válido debido al acarreo	18
	+ 0110	Sumar 6	
0001	**1000**	Número BCD válido	
↓	↓		
1	8		

(c)

0001	0110		16
+0001	0101		+ 15
0010	1011	El grupo de la derecha no es válido (>9), el grupo de la izquierda es válido.	31
	+0110	Sumar 6 al código no válido. Sumar el acarreo, 0001, al siguiente grupo.	
0011	**0001**	Número BCD válido	
↓	↓		
3	1		

(d)

	0110	0111		67
	+ 0101	0011		+53
	1011	1010	Ambos grupos no son válidos (>9)	120
	+ 0110	+0110	Sumar 6 a ambos grupos	
0001	**0010**	**0000**	Número BCD válido	
↓	↓	↓		
1	2	0		

Problema relacionado Sumar los números BCD: 01001000 + 00110100.

REVISIÓN DE LA SECCIÓN 2.10

1. ¿Cuál es el peso binario de los siguientes números BCD?
   (a) 0010 **(b)** 1000 **(c)** 0001 **(d)** 0100
2. Convertir a BCD los siguientes números decimales:
   (a) 6 **(b)** 15 **(c)** 273 **(d)** 849
3. ¿Qué números decimales representan los siguientes códigos?
   (a) 10001001 **(b)** 001001111000 **(c)** 000101010111
4. En la suma BCD, ¿cuándo no es válida una suma de 4 bits?

2.11 CÓDIGOS DIGITALES

Existen muchos códigos especializados que se usan en los sistemas digitales. Acaba de aprender el código BCD, ahora vamos a ver algunos otros. Algunos códigos son estrictamente numéricos, como BCD, y otros son alfanuméricos; es decir, se utilizan para representar números, letras, símbolos e instrucciones. En esta sección se presentan el código Gray y el código ASCII.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Explicar la ventaja del código Gray. ■ Convertir entre código Gray y código binario. ■ Utilizar el código ASCII.

El código Gray

▲ *La característica del código Gray de que sólo cambie un bit entre números sucesivos minimiza las probabilidades de error.*

El **código Gray** es un código sin pesos y no aritmético; es decir, no existen pesos específicos asignados a las posiciones de los bits. La característica más importante del código Gray es que *sólo varía un bit de un código al siguiente*. Esta propiedad es importante en muchas aplicaciones, tales como los codificadores de eje de posición, en los que la susceptibilidad de error aumenta con el número de cambios de bit entre números adyacentes dentro de una secuencia.

La Tabla 2.6 presenta el código Gray de cuatro bits para los números decimales de 0 a 15. Como referencia se muestran también en la tabla los números binarios. Como en los números binarios, *el código Gray puede tener cualquier número de bits*. Observe que, en este código, sólo cambia un bit entre los sucesivos números. Por ejemplo, para pasar del decimal 3 al 4, el código Gray lo hace de 0010 a 0110, mientras que el código binario lo hace de 0011 a 0100, cambiando tres bits. En el código Gray, el único bit que cambia es el tercer bit de la derecha y los restantes permanecen igual.

Conversión de código binario a código Gray. Algunas veces, la conversión de código binario a código Gray resulta útil. Las siguientes reglas explican cómo convertir un número binario en un número en código Gray:

1. El bit más significativo (el que está más a la izquierda, MSB) en el código Gray es el mismo que el correspondiente MSB del número binario.
2. Yendo de izquierda a derecha, sumar cada par adyacente de los bits en código binario para obtener el siguiente bit en código Gray. Los acarreos deben descartarse.

Por ejemplo, la conversión del número binario 10110 a código Gray se hace del siguiente modo:

```
1—+→0—+→1—+→1—+→0   Binario
↓     ↓     ↓     ↓     ↓
1     1     1     0     1   Gray
```

El código Gray es 11101.

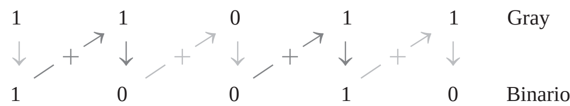
Decimal	Binario	Código Gray	Decimal	Binario	Código Gray
0	0000	0000	8	1000	1100
1	0001	0001	9	1001	1101
2	0010	0011	10	1010	1111
3	0011	0010	11	1011	1110
4	0100	0110	12	1100	1010
5	0101	0111	13	1101	1011
6	0110	0101	14	1110	1001
7	0111	0100	15	1111	1000

TABLA 2.6 Código Gray de cuatro bits.

Conversión de Gray a binario. Para convertir de código Gray a binario, se utiliza un método similar, pero con algunas diferencias. Se aplican las siguientes reglas:

1. El bit más significativo (bit más a la izquierda) en el código binario es el mismo que el correspondiente bit en código Gray.
2. A cada bit del código binario generado se le suma el bit en código Gray de la siguiente posición adyacente. Los acarreos se descartan.

Por ejemplo, la conversión del número en código Gray 11011 a binario es como sigue:

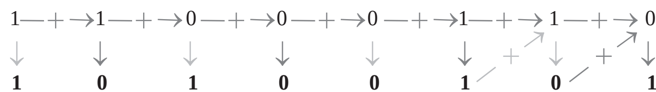


El número binario es 10010.

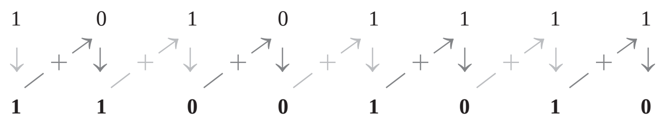
EJEMPLO 2.37

(a) Convertir el número binario 11000110 a código Gray.
(b) Convertir a binario el código Gray 10101111.

Solución **(a)** Binario a código Gray:



(b) Código Gray a binario:



Problema relacionado **(a)** Convertir a código Gray el número binario 101101. **(b)** Convertir a binario el código Gray 100111.

Una aplicación

En la Figura 2.7 se muestra un diagrama simplificado de un mecanismo codificador de eje de 3 bits. Básicamente, está formado por tres anillos concéntricos conductores que están divididos en ocho sectores. Cuantos más sectores haya, con mayor precisión se puede representar la posición, pero aquí sólo se utilizan ocho con propósitos de ilustración. Cada sector de cada anillo se fija a un nivel alto o a un nivel bajo de tensión para representar un 1 o un 0. Los sectores de color gris indican los 1s, y los sectores blancos los 0s. Puesto que los anillos giran con el eje, hacen contacto con una disposición de escobillas que están en una posición fija, y a las que están conectadas las líneas de salida. Cuando el eje gira en el sentido contrario a las agujas del reloj a través de los 360º, los ocho sectores se mueven y pasan por las tres escobillas produciendo una salida binaria de tres bits que indica la posición del eje.

En la Figura 2.7(a), los sectores están colocados según el modelo binario normal, por lo que las escobillas van de 000 a 001, de 010 a 011, y así sucesivamente. Cuando las escobillas se encuentran sobre los sectores de color gris, la salida es 1, y cuando están sobre los sectores blancos, la salida es 0. Si una de las escobillas, durante la transición de un sector al siguiente, se adelanta ligeramente, puede producirse una salida errónea. Consideremos lo que ocurre cuando las escobillas están en el sector 111, y pasan al sector 000. Si la escobilla MSB se adelanta ligeramente, la posición se indicaría de forma incorrecta por una transición a 011 en lugar de 111 a 000. En este tipo de aplicaciones, es virtualmente imposible mantener el alineamiento mecánico preciso de todas las escobillas, por lo que siempre se producirán algunos errores en muchas de las transiciones entre sectores.

El código Gray se utiliza para eliminar los errores que son inherentes al código binario. Como se muestra en la Figura 2.7(b), el código Gray asegura que sólo un bit variará entre sectores adyacentes. Esto significa que aunque las escobillas no puedan tener un alineamiento preciso, nunca se producirá un error de transición. Por ejemplo, consideremos de nuevo qué ocurre cuando las escobillas están en el sector 111 y se mueven al sector siguiente, 101. Las dos únicas posibles salidas durante la transición son 111 y 101, independientemente de cómo estén alineadas las escobillas. Una situación similar se produce en las transiciones entre los restantes sectores.

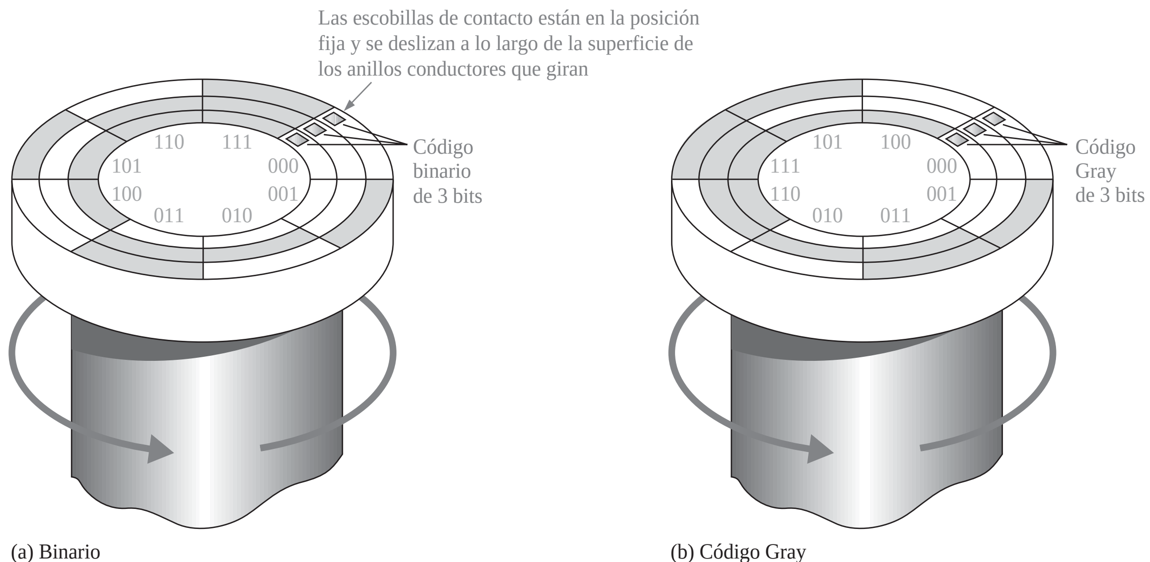


FIGURA 2.7 Ilustración simplificada de cómo el código Gray resuelve el problema de error en los codificadores de posición de eje.

Códigos alfanuméricos

Para la comunicación, no sólo se necesitan números, sino también letras y otros símbolos. En sentido estricto, los códigos **alfanuméricos** son códigos que representan números y caracteres alfabéticos (letras). Sin embargo, la mayoría de estos códigos también representan otros caracteres tales como símbolos y distintas instrucciones necesarias para la transferencia de información.

Como mínimo, un código alfanumérico debe poder representar los diez dígitos decimales y las 26 letras del alfabeto, es decir, un total de 36 elementos. Esta cantidad requiere seis bits para cada combinación de código, puesto que cinco son insuficientes ($2^5 = 32$). Con seis bits se tiene un total de 64 combinaciones, por lo que 28 de ellas no se utilizan. Obviamente, en muchas aplicaciones, para completar la comunicación, son necesarios otros símbolos además de los números y las letras. Se necesitan espacios, punto, dos puntos, punto y coma, signo de interrogación, etcétera. También se necesitan instrucciones para comunicar al sistema receptor qué hacer con la información. De este modo, con códigos con una longitud de seis bits, se pueden manejar números decimales, el alfabeto y otros 28 símbolos. Esto debería darle una idea de los requerimientos para un código alfanumérico básico. El ASCII es el código alfanumérico más común y se va a tratar a continuación.

ASCII

El ***American Standard Code for Information Interchange*** (ASCII, código estándar americano para el intercambio de información) es un código alfanumérico universalmente aceptado, que se usa en la mayoría de las computadoras y otros equipos electrónicos. La mayor parte de los teclados de computadora se estandarizan de acuerdo con el código ASCII, y cuando se pulsa una letra, un número o un comando de control, es el código ASCII el que se introduce en la computadora.

El código ASCII dispone de 128 caracteres que se representan mediante un código binario de 7 bits. Realmente, el código ASCII puede considerarse como un código de 8 bits en el que el MSB siempre es 0. En hexadecimal, este código de 8 bits va de 00 hasta 7F. Los primeros 32 caracteres ASCII son comandos no gráficos, que nunca se imprimen o presentan en pantalla, y sólo se utilizan para propósitos de control. Ejemplos de caracteres de control son el carácter "nulo", "avance de línea", "inicio de texto" y "escape". Los demás caracteres son símbolos gráficos que pueden imprimirse o mostrarse en pantalla, e incluyen las letras del alfabeto (mayúsculas y minúsculas), los diez dígitos decimales, los signos de puntuación y otros símbolos comúnmente utilizados.

La Tabla 2.7 presenta un listado del código ASCII, con su representación decimal, hexadecimal y binaria para cada carácter y símbolo. En la primera columna de la tabla se enumeran los nombres de los 32 caracteres de control (en hexadecimal, de 00 hasta 1F), y en las restantes columnas se muestran los símbolos gráficos (en hexadecimal, de 20 hasta 7F).



NOTAS INFORMÁTICAS

Un teclado de computadora tiene un microprocesador dedicado, que explora constantemente los circuitos de teclado para detectar cuándo se ha presionado y liberado una tecla. Un paquete de software genera un código de exploración unívoco que representa a dicha tecla en particular. El código de exploración se convierte a continuación en un código alfanumérico (ASCII) que utilizará la computadora.

Los caracteres de control ASCII. Los primeros treinta y dos códigos de la tabla ASCII (Tabla 2.7) representan los caracteres de control. Estos se utilizan para permitir a dispositivos, tales como una computadora o una impresora, que se comuniquen entre sí cuando transfieren información y datos. La Tabla 2.8 enumera los caracteres de control y las funciones de las teclas de control que permiten introducir directamente el código ASCII a través del teclado, presionando la tecla control (CTRL) y el símbolo correspondiente. También se facilita una breve descripción de cada carácter de control.

Caracteres de control				Símbolos gráficos											
Nombre	**Dec**	**Binario**	**Hex**	**Símbolo**	**Dec**	**Binario**	**Hex**	**Nombre**	**Dec**	**Binario**	**Hex**	**Símbolo**	**Dec**	**Binario**	**Hex**
NUL	0	0000000	00	space	32	0100000	20	@	64	1000000	40	`	96	1100000	60
SOH	1	0000001	01	!	33	0100001	21	A	65	1000001	41	a	97	1100001	61
STX	2	0000010	02	“	34	0100010	22	B	66	1000010	42	b	98	1100010	62
ETX	3	0000011	03	#	35	0100011	23	C	67	1000011	43	c	99	1100011	63
EOT	4	0000100	04	$	36	0100100	24	D	68	1000100	44	d	100	1100100	64
ENQ	5	0000101	05	%	37	0100101	25	E	69	1000101	45	e	101	1100101	65
ACK	6	0000110	06	&	38	0100110	26	F	70	1000110	46	f	102	1100110	66
BEL	7	0000111	07	’	39	0100111	27	G	71	1000111	47	g	103	1100111	67
BS	8	0001000	08	(	40	0101000	28	H	72	1001000	48	h	104	1101000	68
HT	9	0001001	09	)	41	0101001	29	I	73	1001001	49	i	105	1101001	69
LF	10	0001010	0A	*	42	0101010	2A	J	74	1001010	4A	j	106	1101010	6A
VT	11	0001011	0B	+	43	0101011	2B	K	75	1001011	4B	k	107	1101011	6B
FF	12	0001100	0C	’	44	0101100	2C	L	76	1001100	4C	l	108	1101100	6C
CR	13	0001101	0D	–	45	0101101	2D	M	77	1001101	4D	m	109	1101101	6D
SO	14	0001110	0E	.	46	0101110	2E	N	78	1001110	4E	n	110	1101110	6E
SI	15	0001111	0F	/	47	0101111	2F	O	79	1001111	4F	o	111	1101111	6F
DLE	16	0010000	10	0	48	0110000	30	P	80	1010000	50	p	112	1110000	70
DC1	17	0010001	11	1	49	0110001	31	Q	81	1010001	51	q	113	1110001	71
DC2	18	0010010	12	2	50	0110010	32	R	82	1010010	52	r	114	1110010	72
DC3	19	0010011	13	3	51	0110011	33	S	83	1010011	53	s	115	1110011	73
DC4	20	0010100	14	4	52	0110100	34	T	84	1010100	54	t	116	1110100	74
NAK	21	0010101	15	5	53	0110101	35	U	85	1010101	55	u	117	1110101	75
SYN	22	0010110	16	6	54	0110110	36	V	86	1010110	56	v	118	1110110	76
ETB	23	0010111	17	7	55	0110111	37	W	87	1010111	57	w	119	1110111	77
CAN	24	0011000	18	8	56	0111000	38	X	88	1011000	58	x	120	1111000	78
EM	25	0011001	19	9	57	0111001	39	Y	89	1011001	59	y	121	1111001	79
SUB	26	0011010	1A	:	58	0111010	3A	Z	90	1011010	5A	z	122	1111010	7A
ESC	27	0011011	1B	;	59	0111011	3B	[	91	1011011	5B	{	123	1111011	7B
FS	28	0011100	1C	<	60	0111100	3C	\	92	1011100	5C	|	124	1111100	7C
GS	29	0011101	1D	=	61	0111101	3D	]	93	1011101	5D	}	125	1111101	7D
RS	30	0011110	1E	>	62	0111110	3E	^	94	1011110	5E	~	126	1111110	7E
US	31	0011111	1F	?	63	0111111	3F	_	95	1011111	5F	Del	127	1111111	7F

TABLA 2.7 Código ASCII (American Standard Code for Information Interchange).

EJEMPLO 2.38

Determinar los códigos binarios ASCII que se han introducido a través del teclado de la computadora cuando se ha escrito la instrucción BASIC siguiente. Expresar también cada código en hexadecimal.

20 PRINT “A=”;X

Solución En la Tabla 2.7 puede encontrar el código ASCII correspondiente a cada carácter.

Símbolo	Binario	Hexadecimal
2	0110010	32_{16}
0	0110000	30_{16}
Space	0100000	20_{16}
P	1010000	50_{16}
R	1010010	52_{16}
I	1001001	49_{16}
N	1001110	$4E_{16}$
T	1010100	54_{16}
Space	0100000	20_{16}
"	0100010	22_{16}
A	1000001	41_{16}
=	0111101	$3D_{16}$
"	0100010	22_{16}
;	0111011	$3B_{16}$
X	1011000	58_{16}

Problema relacionado Determinar la secuencia de los códigos ASCII requerida para la siguiente instrucción de programa y expresarla en hexadecimal:

80 INPUT Y

Caracteres del código ASCII extendido

Además de los 128 caracteres ASCII estándar, existen 128 caracteres adicionales que fueron adoptados por IBM para utilizar en sus computadoras personales (PC). Debido a la popularidad del PC, estos caracteres especiales del código ASCII extendido se usan también en otras aplicaciones distintas de los PC, por lo que se ha convertido en un estándar no oficial.

Los caracteres del código ASCII extendido se representan mediante una serie de códigos de 8 bits que van, en hexadecimal, del 80 hasta FF.

El código ASCII extendido está formado por caracteres que pertenecen a las siguiente categorías generales:

1. Caracteres alfabéticos no ingleses.
2. Símbolos de moneda no ingleses.
3. Letras griegas
4. Símbolos matemáticos

Nombre	Decimal	Hex	Tecla	Descripción
NUL	0	00	CTRL @	Carácter nulo
SOH	1	01	CTRL A	Inicio de cabecera
STX	2	02	CTRL B	Inicio de texto
ETX	3	03	CTRL C	Fin de texto
EOT	4	04	CTRL D	Fin de transmisión
ENQ	5	05	CTRL E	Petición
ACK	6	06	CTRL F	Reconocimiento
BEL	7	07	CTRL G	Timbre
BS	8	08	CTRL H	Barra espaciadora
HT	9	09	CTRL I	Tabulador horizontal
LF	10	0A	CTRL J	Avance de línea
VT	11	0B	CTRL K	Tabulador vertical
FF	12	0C	CTRL L	Salto de página
CR	13	0D	CTRL M	Retorno de carro
SO	14	0E	CTRL N	Desplazamiento de salida
SI	15	0F	CTRL O	Desplazamiento de entrada
DLE	16	10	CTRL P	Escape de enlace de datos
DC1	17	11	CTRL Q	Dispositivo de control 1
DC2	18	12	CTRL R	Dispositivo de control 2
DC3	19	13	CTRL S	Dispositivo de control 3
DC4	20	14	CTRL T	Dispositivo de control 4
NAK	21	15	CTRL U	Confirmación negativa
SYN	22	16	CTRL V	Sincronismo
ETB	23	17	CTRL W	Fin del bloque de transmisión
CAN	24	18	CTRL X	Cancelación
EM	25	19	CTRL Y	Fin del dispositivo
SUB	26	1A	CTRL Z	Sustitución
ESC	27	1B	CTRL [	Escape
FS	28	1C	CTRL /	Separador de archivo
GS	29	1D	CTRL]	Separador de grupo
RS	30	1E	CTRL ^	Separador de registro
US	31	1F	CTRL _	Separador de unidad

TABLA 2.8 Caracteres de control ASCII.

5. Caracteres para gráficos
6. Caracteres para gráficos de barras
7. Caracteres sombreados

En la Tabla 2.9 se enumera el conjunto de caracteres del código ASCII extendido, junto con sus representaciones decimal y hexadecimal.

Símbolo	Dec	Hex	Símbolo	Dec	Hex	Símbolo	Dec	Hex	Símbolo	Dec	Hex
Ç	128	80	á	160	A0	└	192	C0	α	224	E0
ü	129	81	í	161	A1	┴	193	C1	β	225	E1
é	130	82	ó	162	A2	┬	194	C2	Γ	226	E2
â	131	83	ú	163	A3	├	195	C3	π	227	E3
ä	132	84	ñ	164	A4	─	196	C4	Σ	228	E4
à	133	85	Ñ	165	A5	┼	197	C5	σ	229	E5
å	134	86	ª	166	A6	╞	198	C6	µ	230	E6
ç	135	87	º	167	A7	╟	199	C7	τ	231	E7
ê	136	88	¿	168	A8	╚	200	C8	Φ	232	E8
ë	137	89	⌐	169	A9	╔	201	C9	Θ	233	E9
è	138	8A	¬	170	AA	╩	202	CA	Ω	234	EA
ï	139	8B	½	171	AB	╦	203	CB	δ	235	EB
î	140	8C	¼	172	AC	╠	204	CC	∞	236	EC
ì	141	8D	¡	173	AD	═	205	CD	φ	237	ED
Ä	142	8E	«	174	AE	╬	206	CE	ε	238	EE
Å	143	8F	»	175	AF	╧	207	CF	∩	239	EF
É	144	90	░	176	B0	╨	208	D0	≡	240	F0
æ	145	91	▒	177	B1	╤	209	D1	±	241	F1
Æ	146	92	▓	178	B2	╥	210	D2	≥	242	F2
ô	147	93	│	179	B3	╙	211	D3	≤	243	F3
ö	148	94	┤	180	B4	╘	212	D4	⌠	244	F4
ò	149	95	╡	181	B5	╒	213	D5	⌡	245	F5
û	150	96	╢	182	B6	╓	214	D6	÷	246	F6
ù	151	97	╖	183	B7	╫	215	D7	≈	247	F7
ÿ	152	98	╕	184	B8	╪	216	D8	°	248	F8
Ö	153	99	╣	185	B9	┘	217	D9	•	249	F9
Ü	154	9A	║	186	BA	┌	218	DA	·	250	FA
¢	155	9B	╗	187	BB	█	219	DB	√	251	FB
£	156	9C	╝	188	BC	▄	220	DC	η	252	FC
¥	157	9D	╜	189	BD	▌	221	DD	²	253	FD
₧	158	9E	╛	190	BE	▐	222	DE	■	254	FE
ƒ	159	9F	┐	191	BF	▀	223	DF	□	255	FF

TABLA 2.9 Caracteres ASCII extendidos.

REVISIÓN DE LA SECCIÓN 2.11

1. Convertir a código Gray los siguientes números binarios:
   (a) 1100 **(b)** 1010 **(c)** 11010
2. Convertir a binario los siguientes códigos Gray:
   (a) 1000 **(b)** 1010 **(c)** 11101
3. ¿Cuál es la representación ASCII de cada uno de los caracteres siguientes?

Expresarlos como un patrón de bits y en notación hexadecimal.

(a) K **(b)** r **(c)** $ **(d)** +

2.12 DETECCIÓN DE ERRORES Y CÓDIGOS DE CORRECIÓN

En esta sección se abordan dos métodos para sumar bits a códigos para detectar o para detectar y corregir un error de un único bit. Se presenta el método de paridad para la detección de errores y el método Hamming para detección y corrección de un único error. Cuando se detecta que un bit es erróneo en una determinada palabra de código, puede corregirse simplemente invirtiéndolo.

Al finalizar esta sección, el lector será capaz de:

■ Determinar si existe un error en un código basándose en el bit de paridad. ■ Asignar el apropiado bit de paridad a un código. ■ Utilizar el código Hamming para la detección y corrección de un único error. ■ Asignar los apropiados bits de paridad para corregir un único error.

Método de paridad para la detección de errores

▲ *Un bit de paridad indica si el número de 1s es impar o par.*

Muchos sistemas emplean un bit de paridad como medio para la **detección de errores** de bit. Cualquier grupo de bits contiene un número par o impar de 1s. Un bit de paridad se añade al grupo de bits para hacer que el número total de 1s en el grupo sea siempre par o siempre impar. Un bit de paridad par hace que el número total de 1s sea par, y un bit de paridad impar hace que el número total de 1s del grupo sea impar.

Un determinado sistema puede funcionar con **paridad** par o impar, pero no con ambas. Por ejemplo, si un sistema trabaja con paridad par, una comprobación que se realice en cada grupo de bits recibidos tiene que asegurar que el número total de 1s en ese grupo es par. Si hay un número impar de 1s, quiere decir que se ha producido un error.

Vamos a ver cómo se asocian los bits de paridad a un código. La Tabla 2.10 enumera los bits de paridad de cada número BCD, tanto en el caso de paridad par como de paridad impar. El bit de paridad para cada número BCD se indica en la columna *P*.

El bit de paridad se puede añadir al principio o al final del código, dependiendo del diseño del sistema. Observe que el número total de 1s, incluyendo el bit de paridad, siempre es par para paridad par, y siempre es impar para paridad impar.

Paridad par		Paridad impar	
P	**BCD**	***P***	**BCD**
0	0000	1	0000
1	0001	0	0001
1	0010	0	0010
0	0011	1	0011
1	0100	0	0100
0	0101	1	0101
0	0110	1	0110
1	0111	0	0111
1	1000	0	1000
0	1001	1	1001

TABLA 2.10 El código BCD con bits de paridad.

Detección de un error. Un bit de paridad facilita la detección de un único error de bit (o de cualquier número impar de errores, lo cual es muy improbable), pero no puede detectar dos errores dentro de un grupo. Por ejemplo, supongamos que deseamos transmitir el código BCD 0101 (el método de paridad puede usarse con cualquier número de bits, ahora usamos cuatro con propósitos de ilustración). El código total transmitido incluyendo el bit de paridad par es:

Bit de paridad par

0 0101

Código BCD

Supongamos ahora que se produce un error en el tercer bit de la izquierda (el 1 se transmite como 0).

Bit de paridad par

00001

Bit erróneo

Cuando se recibe este código, la circuitería de comprobación de paridad determina que sólo hay un 1 (impar), cuando debería haber un número par de 1s. Puesto que en el código recibido no aparece un número par de 1s, esto indica que se ha producido un error.

Un bit de paridad impar también facilita de forma similar la detección de un único error en un grupo de bits dado.

EJEMPLO 2.39

Asignar el bit de paridad par apropiado a cada uno de los siguientes grupos de códigos:

(a) 1010 **(b)** 111000 **(c)** 101101
(d) 1000111001001 **(e)** 101101011111

Solución El bit de paridad puede ser 1 o 0 de modo que el número total de 1s sea par. El bit de paridad será el bit más a la izquierda (tramado).

(a) 01010 **(b) 1**111000 **(c) 0**101101
(d) 01000111001001 **(e) 1**101101011111

Problema relacionado añadir un bit de paridad par al código ASCII de 7 bits correspondiente a la letra K.

EJEMPLO 2.40

Un sistema de paridad impar recibe los siguientes grupos de códigos: 10110, 11010, 110011, 110101110100 y 1100010101010. Determinar qué grupos, si hay alguno, tienen error.

Solución Puesto que se requiere paridad impar, cualquier grupo con un número par de 1s es incorrecto. Los siguientes grupos contienen error:

110011 y **1100010101010.**

Problema relacionado En un sistema de paridad impar se recibe el siguiente carácter ASCII: 00110111. ¿Es correcto?

El código Hamming de corrección de errores

Como hemos visto, un único bit de paridad permite detectar errores de un único bit en una palabra de código. Un único bit de paridad puede indicar si existe un error en un determinado grupo de bits. Para corregir un error detectado, se necesita más información, ya que hay que identificar la posición del bit erróneo antes de poder corregirlo. Debe incluirse más de un bit de paridad en un grupo de bits para poder corregir el error detectado. En un código de 7 bits, existen siete posibles bits erróneos. En este caso, tres bits de paridad no sólo pueden detectar el error sino que también pueden especificar la posición del bit erróneo. El **código Hamming** proporciona un método de corrección de un único bit erróneo. A continuación se estudia la construcción de un código Hamming de 7 bits para corregir un único error.

Número de bits de paridad. Si el número de bits de datos se designa por d, entonces el número de bits de paridad, p, se determina mediante la siguiente relación:

Ecuación 2.1 $2^p \geq d + p + 1$

Por ejemplo, si tenemos cuatro bits de datos, entonces p se calcula por el método de prueba y error usando la Ecuación 2.1. Sea $p = 2$. Entonces,

$$2^p = 2^2 = 4$$

y

$$d + p + 1 = 4 + 2 + 1 = 7$$

Puesto que 2^p tiene que ser igual o mayor que $d + p + 1$, la relación de la Ecuación 2.1 *no* se satisface. Probamos de nuevo, sea $p = 3$. Luego,

$$2^p = 2^3 = 8$$

y

$$d + p + 1 = 4 + 3 + 1 = 8$$

Este valor de p satisface la relación de la Ecuación 2.1, por lo que se necesitan tres bits de paridad para poder corregir un único error en cuatro bits de datos. Debemos destacar que se proporciona la detección y corrección de errores para *todos* los bits, tanto de paridad como de datos, del grupo de códigos; es decir, los bits de paridad también se comprueban a sí mismos.

Colocación de los bits de paridad en el código. Ahora que ya sabemos cuál es el número necesario de bits de paridad en nuestro ejemplo, debemos colocar correctamente los bits dentro del código. Debe darse cuenta de que, en este ejemplo, el código está formado por cuatro bits de datos y tres bits de paridad. El bit más a la izquierda es el *bit 1*, el siguiente bit es el *bit 2*, y así sucesivamente, como se muestra a continuación:

bit 1, bit 2, bit 3, bit 4, bit 5, bit 6, bit 7

Los bits de paridad se sitúan en las posiciones que se han numerado haciéndolas corresponder con las potencias de dos en sentido ascendente (1, 2, 4, 8, . . .), del modo siguiente:

$P_1, P_2, D_1, P_3, D_2, D_3, D_4$

El símbolo P_n designa un determinado bit de paridad y D_n designa cada uno de los bits de datos.

Asignación de los valores de los bits de paridad. Para terminar, hay que asignar apropiadamente un valor de 1 o de 0 a cada uno de los bits de paridad. Puesto que cada bit de paridad proporciona una comprobación sobre los restantes bits del código total, tenemos que conocer el valor de dichos otros bits par asignar el valor del bit de paridad. Para hallar los valores de los bits, primero expresamos en binario el número correspondiente a cada posición de bit; es decir, escribimos el número binario correspondiente a cada número decimal de posi-

ción, como se muestra en la Tabla 2.11. A continuación, como se ilustra en la primera fila de la Tabla 2.11, indicamos las posiciones de los bits de paridad y de datos. Observe que el número binario de posición del bit de paridad P_1 tiene un 1 como su dígito más a la derecha. *Este bit de paridad comprueba las posiciones de todos los bits, incluyéndose a sí mismo, que tienen 1s en la misma posición en el correspondiente número de posición en binario*. Por tanto, el bit de paridad P_1 comprueba las posiciones de bits 1, 3, 5 y 7.

Designación de bit	P_1	P_2	D_1	P_3	D_2	D_3	D_4
Posición de bit	1	2	3	4	5	6	7
Número de posición en binario	001	010	011	100	101	110	111
Bits de datos (D_n)							
Bits de paridad (P_n)							

TABLA 2.11 Tabla de posiciones de bits para un código de corrección de errores de 7 bits.

El número de posición en binario para el bit de paridad P_2 tiene un 1 en su posición intermedia. Este bit comprueba entonces todas las posiciones de bit, incluyéndose a sí mismo, que tienen un 1 en esa misma posición. Por tanto, el bit de paridad P_2 comprueba las posiciones de bit 2, 3, 6 y 7.

El número de posición en binario para el bit de paridad P_3 tiene un 1 como su bit más a la izquierda. Este bit comprueba entonces todas las posiciones de bit, incluyéndose a sí mismo, que tienen un 1 en esa misma posición. Por tanto, el bit de paridad P_3 comprueba las posiciones de bit 4, 5, 6 y 7.

En cada uno de los casos, se asigna un valor al bit de paridad de modo que la cantidad de 1s en el conjunto de bits que se desea comprobar sea impar o par, dependiendo de lo que se haya especificado. Los siguientes ejemplos clarificarán este procedimiento.

EJEMPLO 2.41

Determinar el código Hamming para el número BCD 1001 (bits de datos), utilizando paridad par.

Solución

Paso 1. Hallar el número de bits de paridad requeridos. Sea $p = 3$. Entonces,

$$2^p = 2^3 = 8$$

$$d + p + 1 = 4 + 3 + 1 = 8$$

Tres bits de paridad son suficientes.

Número total de bits de código = 4 + 3 = 7

Paso 2. Construir la tabla de posiciones de los bits, como se muestra en la Tabla 2.12, e introducir los bits de datos. Los bits de paridad se determinan en los pasos siguientes.

Designación de bit	P_1	P_2	D_1	P_3	D_2	D_3	D_4
Posición de bit	1	2	3	4	5	6	7
Número de posición en binario	001	010	011	100	101	110	111
Bits de datos			1		0	0	1
Bits de paridad	0	0		1			

Tabla 2.12

Paso 3. Determinar los bits de paridad como sigue:
El bit P_1 comprueba las posiciones de bit 1, 3, 5 y 7, y debe ser igual a 0 para que haya un número par de 1s (2) en este grupo.
El bit P_2 comprueba las posiciones de bit 2, 3, 6 y 7, y debe ser igual a 0 para que haya un número par de 1s (2) en este grupo.
El bit P_3 comprueba las posiciones de bit 4, 5, 6 y 7, y debe ser igual a 1 para que haya un número par de 1s (2) en este grupo.

Paso 4. Estos bits de paridad se anotan en la Tabla 2.12 y el código combinado resultante es 0011001.

Problema relacionado Determinar el código Hamming para el número BCD 1000 utilizando paridad par.

EJEMPLO 2.42

Determinar el código Hamming para los bits de datos 10110 utilizando paridad impar.

Solución

Paso 1. Determinar el número de bits de paridad necesario. En este caso, el número de bits de datos, d, es cinco. Del ejemplo anterior sabemos que $p = 3$ no es apropiado, por lo que probamos $p = 4$:

$$2^p = 2^4 = 16$$
$$d + p + 1 = 5 + 4 + 1 = 10$$

Cuatro bits de paridad son suficientes.

Número total de bits de código = 5 + 4 = 9

Paso 2. Construir una tabla de posiciones de los bits, Tabla 2.13, y escribir los bits de datos. Los bits de paridad se determinan en los pasos siguientes. Observe que P_4 se encuentra en la posición de bit 8.

Designación de bit **Posición de bit** **Número de posición en binario**	P_1 1 0001	P_2 2 0010	D_1 3 0011	P_3 4 0100	D_2 5 0101	D_3 6 0110	D_4 7 0111	P_4 8 1000	D_5 9 1001
Bits de datos			1		0	1	1		0
Bits de paridad	1	0		1				1	

Tabla 2.13

Paso 3. Determinar los bits de paridad como sigue:
El bit P_1 comprueba las posiciones de bit 1, 3, 5, 7 y 9, y debe ser igual a 1 para que haya un número impar de 1s (3) en este grupo.
El bit P_2 comprueba las posiciones de bit 2, 3, 6 y 7, y debe ser igual a 0 para que haya un número impar de 1s (3) en este grupo.
El bit P_3 comprueba las posiciones de bit 4, 5, 6 y 7, y debe ser igual a 1 para que haya un número impar de 1s (3) en este grupo.
El bit P_4 comprueba las posiciones de bit 8 y 9, y debe ser igual a 1 para que haya un número impar de 1s (1) en este grupo.

Paso 4. Estos bits de paridad se introducen en la Tabla 2.13, y el código combinado resultante es 101101110.

Problema relacionado Determinar el código Hamming para el número 11001 usando paridad impar.

Cómo detectar y corregir un error con el código Hamming

Ahora que ya conoce el método de construcción de un código de corrección de errores, ¿cómo se emplea para localizar y corregir un error? Cada uno de los bits de paridad junto con su correspondiente grupo de bits debe comprobarse de acuerdo con la paridad que se vaya a utilizar. Si en una palabra de código hay tres bits de paridad, entonces se realizan tres comprobaciones de paridad. Si hay cuatro bits de paridad, deben realizarse cuatro comprobaciones, y así sucesivamente. Cada comprobación de paridad dará un resultado bueno o malo. El resultado total de todas las comprobaciones de paridad indica el bit, si existe, en el que se encuentra el error de la siguiente manera:

Paso 1. Comience con el grupo comprobado por P_1.

Paso 2. Compruebe si el grupo tiene la paridad correcta. Un 0 representa que la comprobación de paridad es correcta y un 1 que es incorrecta.

Paso3. Repita el paso 2 para cada grupo de paridad.

Paso 4. El número binario formado por los resultados de todas las comprobaciones de paridad indica la posición del bit del código que es erróneo. Es el *código de posición de error.* La primera comprobación de paridad genera el bit menos significativo (LSB). Si todas las comprobaciones son correctas, no habrá error.

EJEMPLO 2.43

Suponga que se transmite la palabra código del Ejemplo 2.41 (0011001) y que se recibe 0010001. El receptor no "sabe" lo que se ha transmitido y debe calcular las paridades apropiadas para determinar si el código es correcto. Indique cualquier error que se haya producido en la transmisión si se utiliza paridad par.

Solución En primer lugar, construimos una tabla de posiciones de bits, como la mostrada en la Tabla 2.14.

Designación de bit	P_1	P_2	D_1	P_3	D_2	D_3	D_4
Posición de bit	1	2	3	4	5	6	7
Número de posición en binario	001	010	011	100	101	110	111
Código recibido	0	0	1	0	0	0	1

Tabla 2.14

Primera comprobación de paridad:

El bit P_1 comprueba las posiciones 1, 3, 5 y 7.

En este grupo hay dos 1s.

La comprobación de paridad es correcta. ⟶ 0 (LSB)

Segunda comprobación de paridad:
El bit P_2 comprueba las posiciones 2, 3, 6 y 7.
En este grupo hay dos 1s.
La comprobación de paridad es correcta. ⟶ 0
Tercera comprobación de paridad:
El bit P_3 comprueba las posiciones 4, 5, 6 y 7.
En este grupo hay un 1.
La comprobación de paridad es incorrecta. ⟶ 1 (MSB)
Resultado:
El código de posición del error es 100 (cuatro en binario). Esto quiere decir que el bit que se encuentra en la posición 4 es erróneo. Se ha recibido un 0 y tiene que ser un 1. El código corregido es 0011001, que es el mismo que el código transmitido.

Problema relacionado Repita el proceso ilustrado en el ejemplo para el caso de que el código recibido fuera 0111001.

EJEMPLO 2.44

Se recibe el código 101101010. Corregir los errores. Se emplean cuatro bits de paridad y el tipo de paridad impar.

Solución En primer lugar, construimos una tabla de posiciones de bits, como la mostrada en la Tabla 2.15.

Designación de bit	P_1	P_2	D_1	P_3	D_2	D_3	D_4	P_4	D_5
Posición de bit	1	2	3	4	5	6	7	8	9
Número de posición en binario	0001	0010	0011	0100	0101	0110	0111	1000	1001
Código recibido	1	0	1	1	0	1	0	1	0

Tabla 2.15

Primera comprobación de paridad:
El bit P_1 comprueba las posiciones 1, 3, 5, 7 y 9.
En este grupo hay dos 1s.
La comprobación de paridad es incorrecta. ⟶ 1 (LSB)
Segunda comprobación de paridad:
El bit P_2 comprueba las posiciones 2, 3, 6 y 7.
En este grupo hay dos 1s.
La comprobación de paridad es incorrecta. ⟶ 1
Tercera comprobación de paridad:
El bit P_3 comprueba las posiciones 4, 5, 6 y 7.
En este grupo hay un 1.
La comprobación de paridad es incorrecta. ⟶ 1
Cuarta comprobación de paridad:
El bit P_4 comprueba las posiciones 8 y 9.

En este grupo hay un 1.
La comprobación de paridad es correcta. ⟶ 0 (MSB)

Resultado:
El código de posición de error es 0111 (siete en binario). Esto quiere decir que l bit situado en la posición 7 es erróneo. Por tanto, el código corregido es 101101110.

Problema relacionado Se recibe el código 101111001. Corrija cualquier error que se haya producido utilizando paridad impar.

REVISIÓN DE LA SECCIÓN 2.12

1. ¿Qué códigos de paridad impar son erróneos?
(a) 1011 **(b)** 1110 **(c)** 0101 **(d)** 1000

2. ¿Qué códigos de paridad par son erróneos?
(a) 11000110 **(b)** 00101000 **(c)** 10101010 **(d)** 11111011

3. Sumar un bit de paridad par al final de cada uno de los siguientes códigos.
(a) 1010100 **(b)** 0100000 **(c)** 1110111 **(d)** 1000110

4. ¿Cuántos bits de paridad son necesarios para los bits de datos 11010 utilizando el código Hamming?

5. Crear el código Hamming para los bits de datos 0011 utilizando paridad impar.

RESUMEN

- Un número binario es un grupo de bits con peso en el que el peso de cada número entero es una potencia positiva de dos, y el peso de cada dígito fraccionario es una potencia negativa de dos. Los pesos de los números enteros aumentan de derecha a izquierda, del bit menos significativo al más significativo.
- Un número binario puede convertirse a número decimal sumando los valores decimales de los pesos de todos los 1s del número binario.
- Un número entero decimal puede convertirse a binario utilizando la suma de pesos o el método de la división sucesiva por 2.
- Una fracción decimal puede convertirse a binario utilizando la suma de pesos o el método de multiplicación sucesiva por 2.
- Las reglas básicas de la suma binaria son las siguientes:

  $0 + 0 = 0$
  $0 + 1 = 1$
  $1 + 0 = 1$
  $1 + 1 = 10$
- Las reglas básicas de la resta binaria son las siguientes:

  $0 - 0 = 0$
  $1 - 1 = 0$
  $1 - 0 = 1$
  $10 - 1 = 1$
- El complemento a 1 de un número binario se obtiene cambiando los 1s por 0s, y los 0s por 1s.
- El complemento a 2 de un número binario puede obtenerse sumando 1 al complemento a 1.
- La resta binaria puede realizarse mediante sumas, utilizando los métodos de complemento a 1 y complemento a 2.
- Un número binario positivo se representa mediante un bit de signo 0.

- Un número binario negativo se representa mediante un bit de signo 1.
- Para las operaciones aritméticas, los números binarios negativos se representan en complemento a 1 o en complemento a 2.
- En una operación de suma, se puede producir un desbordamiento cuando ambos números son positivos o negativos. Un bit de signo incorrecto en la suma indica que se ha producido un desbordamiento.
- El sistema de numeración hexadecimal está formado por 16 dígitos y caracteres, de 0 hasta 9 y de A hasta F.
- Un dígito hexadecimal se representa mediante un número binario de cuatro bits, y su principal utilidad es simplificar los modelos binarios y hacerlos más fáciles de leer.
- Un número decimal puede convertirse a hexadecimal por el método de la división sucesiva por 16.
- El sistema de numeración octal se forma con ocho dígitos, de 0 hasta 7.
- Un número decimal puede convertirse a octal utilizando el método de la división sucesiva por 8.
- La conversión octal binario se realiza reemplazando cada dígito octal por su equivalente binario de tres bits. Para la conversión binario-octal se realiza el mismo proceso a la inversa.
- Un número decimal se convierte a BCD reemplazando cada dígito decimal por el apropiado código binario de cuatro bits.
- El código ASCII es un código alfanumérico de siete bits, que se utiliza ampliamente en sistemas de computadora para las entradas y salidas de información.
- Se emplea un bit de paridad para detectar un error en un código.
- El código Hamming proporciona un método de detección y corrección de un único error.

PALABRAS CLAVE

Las palabras clave y los términos que se han resaltado en negrita se encuentran en el glosario final del libro.

ACSII *American Standard Code for Information Interchange*, código estándar americano para el intercambio de información; el código alfanumérico más utilizado.
Alfanumérico Que contiene números, letras y otros caracteres.
BCD *Binary Coded Decimal*, código decimal binario, código digital en el que cada dígito decimal, de 0 a 9, se representa mediante un grupo de cuatro bits.
Bit más significativo (MSB, *Most Significant Bit***)** El bit más a la izquierda de un número entero o código binario.
Bit menos significativo (LSB, *Least Significant Bit***)** El bit más a la derecha de un número entero o código binario.
Byte Grupo de ocho bits.
Código Hamming Un tipo de código de detección y corrección de errores.
Hexadecimal Describe un sistema de numeración en base 16.
Número en coma flotante Representación de un número basada en la notación científica, en la que el número consta de un exponente, una mantisa y un signo.
Octal Describe un sistema de numeración en base ocho.
Paridad En relación a los códigos binarios, tener un número par o impar de unos en un grupo de código.

AUTOTEST

Las respuestas se encuentran al final del capítulo

1. $2 \times 10^1 + 8 \times 10^0$ es igual a

(a) 10 **(b)** 280 (c) 2,8 **(d)** 28

2. El número binario 1101 es igual al número decimal

(a) 13 **(b)** 49 **(c)** 11 **(d)** 3

3. El número binario 11011101 es igual al número decimal

(a) 121 **(b)** 221 **(c)** 441 **(d)** 256

4. El número decimal 17 es igual al número binario

(a) 10010 **(b)** 11000 **(c)** 10001 **(d)** 01001

5. El número decimal 175 es igual al número binario

(a) 11001111 **(b)** 10101110 **(c)** 10101111 **(d)** 11101111

6. La suma de 11010 + 01111 es igual a

(a) 101001 **(b)** 101010 **(c)** 110101 **(d)** 101000

7. La diferencia de 110 − 010 es igual a

(a) 001 **(b)** 010 **(c)** 101 **(d)** 100

8. El complemento a 1 de 10111001 es

(a) 01000111 **(b)** 01000110 **(c)** 11000110 **(d)** 10101010

9. El complemento a 2 de 11001000 es

(a) 00110111 **(b)** 00110001 **(c)** 01001000 **(d)** 00111000

10. El número decimal +22 se expresa en complemento a 2 como

(a) 01111010 **(b)** 11111010 **(c)** 01000101 **(d)** 10000101

11. El número decimal −34 se expresa en complemento a 2 como

(a) 01011110 **(b)** 10100010 **(c)** 11011110 **(d)** 01011101

12. Un número binario en coma flotante de simple precisión tiene un total de

(a) 8 bits **(b)** 16 bits **(c)** 24 bits **(d)** 32 bits

13. En el sistema de complemento a 2, el número binario 10010011 es igual al número decimal

(a) −19 **(b)** +109 **(c)** +91 **(d)** −109

14. El número binario 101100111001010100001 puede escribirse en octal como

(a) 5471230_8 **(b)** 5471241_8

(c) 2634521_8 **(d)** 23162501_8

15. El número binario 10001101010001101111 puede escribirse en hexadecimal como

(a) $AD467_{16}$ **(b)** $8C46F_{16}$ **(c)** $8D46F_{16}$ **(d)** $AE46F_{16}$

16. El número binario correspondiente a $F7A9_{16}$ es

(a) 1111011110101001 **(b)** 1110111110101001

(c) 1111111010110001 **(d)** 1111011010101001

17. El número BCD para el decimal 473 es

(a) 111011010 **(b)** 110001110011

(c) 010001110011 **(d)** 010011110011

18. Utilizando la Tabla 2.7, el comando STOP en ASCII es

(a) 1010011101010010011111010000 **(b)** 1010010100110010011101010000

(c) 1001010110110110011101010001 **(d)** 1010011101010010011101100100

19. El código que tiene un error de paridad par es

(a) 1010011 **(b)** 1101000

(c) 1001000 **(d)** 1110111

PROBLEMAS

Las respuestas a los problemas impares se encuentran al final del libro.

SECCIÓN 2.1 Números decimales

1. ¿Cuál es el peso del dígito 6 en cada uno de los siguientes números decimales?

(a) 1386 **(b)** 54,692 **(c)** 671,920

2. Expresar cada una de los siguientes números decimales como una potencia de diez:

(a) 10 **(b)** 100 **(c)** 10.000 **(d)** 1.000.000

3. Hallar el valor de cada dígito en cada uno de los siguientes números decimales:

(a) 471 **(b)** 9.356 **(c)** 125.000

4. ¿Hasta qué número puede contar con cuatro dígitos decimales?

SECCIÓN 2.2 Números binarios

5. Convertir a decimal los siguientes números binarios:

(a) 11 **(b)** 100 **(c)** 111 **(d)** 1000

(e) 1001 **(f)** 1100 **(g)** 1011 **(h)** 1111

6. Convertir a decimal los siguientes números binarios:

(a) 1110 **(b)** 1010 **(c)** 11100 **(d)** 10000

(e) 10101 **(f)** 11101 **(g)** 10111 **(h)** 11111

7. Convertir a decimal los siguientes números binarios:

(a) 110011,11 **(b)** 101010,01 **(c)** 1000001,111

(d) 1111000,101 **(e)** 1011100,10101 **(f)** 1110001,0001

(g) 1011010,1010 **(h)** 1111111,11111

8. ¿Cuál es el mayor número decimal que se puede representar con cada uno de las siguientes cantidades de dígitos binarios (bits)?

(a) dos **(b)** tres **(c)** cuatro **(d)** cinco **(e)** seis

(f) siete **(g)** ocho **(h)** nueve **(i)** diez **(j)** once

9. ¿Cuántos bits se requieren para representar los siguientes números decimales?

(a) 17 **(b)** 35 **(c)** 49 **(d)** 68

(e) 81 **(f)** 114 **(g)** 132 **(h)** 205

10. Generar la secuencia binaria para las siguientes secuencias decimales:

(a) 0 a 7 **(b)** 8 a 15 **(c)** 16 a 31

(d) 32 a 63 **(e)** 64 a 75

SECCIÓN 2.3 Conversión decimal-binario

11. Convertir a binario cada uno de los números decimales indicados usando el método de la suma de pesos:

(a) 10 **(b)** 17 **(c)** 24 **(d)** 48

(e) 61 **(f)** 93 **(g)** 125 **(h)** 186

12. Convertir a binario cada uno de los números decimales fraccionarios indicados usando el método de la suma de pesos:

(a) 0,32 **(b)** 0,246 **(c)** 0,0981

13. Convertir a binario cada uno de los números decimales indicados usando el método de la división sucesiva por 2:

(a) 15 **(b)** 21 **(c)** 28 **(d)** 34

(e) 40 **(f)** 59 **(g)** 65 **(h)** 73

14. Convertir a binario cada uno de los números decimales fraccionarios indicados usando el método de la multiplicación sucesiva por 2:

(a) 0,98 **(b)** 0,347 **(c)** 0,9028

SECCIÓN 2.4 Aritmética binaria

15. Sumar los números binarios:

(a) 11 + 01 **(b)** 10 + 10 **(c)** 101 + 11

(d) 111 + 110 **(e)** 1001 +101 **(f)** 1101 + 1011

16. Realizar la sustracción directa de los siguientes números binarios:

(a) 11 − 1 **(b)** 101 − 100 **(c)** 110 − 101

(d) 1110 − 11 **(e)** 1100 − 1001 **(f)** 11010 − 10111

17. Realizar las siguientes multiplicaciones binarias:

(a) 11 × 11 **(b)** 100 × 10 **(c)** 111 × 101

(d) 1001 × 110 **(e)** 1101 × 1101 **(f)** 1110 × 1101

18. Dividir los números binarios siguientes:

(a) 100 ÷ 10 **(b)** 1001 ÷ 11 **(c)** 1100 ÷ 100

SECCIÓN 2.5 Complemento a 1 y complemento a 2 de los números binarios

19. Determinar el complemento a 1 de los siguientes números binarios:

(a) 101 **(b)** 110 **(c)** 1010

(d) 11010111 **(e)** 1110101 **(f)** 00001

20. Determinar el complemento a 2 de los siguientes números binarios utilizando cualquier método:

(a) 10 **(b)** 111 **(c)** 1001 **(d)** 1101

(e) 11100 **(f)** 10011 **(g)** 10110000 **(h)** 00111101

SECCIÓN 2.6 Números con signo

21. Expresar en formato binario de 8 bits signo-magnitud los siguientes números decimales:

(a) +29 **(b)** −85 **(c)** +100 **(d)** −123

22. Expresar cada número decimal como un número de 8 bits en el sistema de complemento a 1:

(a) −34 **(b)** +57 **(c)** −99 **(d)** _115

23. Expresar cada número decimal como un número de 8 bits en el sistema de complemento a 2:

(a) +12 **(b)** −68 **(c)** +101 **(d)** −125

24. Determinar el valor decimal de cada número binario con signo en el formato signo-magnitud:

(a) 10011001 **(b)** 01110100 **(c)** 10111111

25. Determinar el valor decimal de cada número binario con signo en el formato de complemento a 1:

(a) 10011001 **(b)** 01110100 **(c)** 10111111

26. Determinar el valor decimal de cada número binario con signo en el formato de complemento a 2:

(a) 10011001 **(b)** 01110100 **(c)** 10111111

27. Expresar cada uno de los siguientes números binarios en formato signo-magnitud en formato de coma flotante de simple precisión:

(a) 0111110000101011

(b) 100110000011000

28. Determinar los valores de los siguientes números en coma flotante de simple precisión:

(a) 1 10000001 01001001110001000000000

(b) 0 11001100 10000111110100100000000

SECCIÓN 2.7 **Operaciones aritméticas de números con signo**

29. Convertir a binario cada pareja de números decimales y sumarlos usando el sistema de complemento a 2:

(a) 33 y 15 **(b)** 56 y –27 **(c)** –46 y 25 **(d)** –110 y –84

30. Realizar las siguientes sumas utilizando el sistema de complemento a 2:

(a) 00010110 + 00110011 **(b)** 01110000 + 10101111

31. Realizar las siguientes sumas utilizando el sistema de complemento a 2:

(a) 10001100 + 00111001 **(b)** 11011001 + 11100111

32. Realizar las siguientes restas utilizando el sistema de complemento a 2:

(a) 00110011 – 00010000 **(b)** 01100101 – 11101000

33. Multiplicar 01101010 por 11110001 utilizando el sistema de complemento a 2.

34. Dividir 01000100 entre 00011001 utilizando el sistema de complemento a 2.

SECCIÓN 2.8 **Números hexadecimales**

35. Convertir a binario los siguientes números hexadecimales:

(a) 38_{16} **(b)** 59_{16} **(c)** $A14_{16}$ **(d)** $5C8_{16}$

(e) 4100_{16} **(f)** $FB17_{16}$ **(g)** $8A9D_{16}$

36. Convertir a hexadecimal los siguientes números binarios:

(a) 1110 **(b)** 10 **(c)** 10111

(d) 10100110 **(e)** 1111110000 **(f)** 100110000010

37. Convertir a decimal los siguientes números hexadecimales:

(a) 23_{16} **(b)** 92_{16} **(c)** $1A_{16}$ **(d)** $8D_{16}$

(e) $F3_{16}$ **(f)** EB_{16} **(g)** $5C2_{16}$ **(h)** 700_{16}

38. Convertir a decimal los siguientes números hexadecimales:

(a) 8 **(b)** 14 **(c)** 33 **(d)** 52

(e) 284 **(f)** 2890 **(g)** 4019 **(h)** 6500

39. Realizar las siguientes sumas:

(a) $37_{16} + 29_{16}$ **(b)** $A0_{16} + 6B_{16}$ **(c)** $FF_{16} + BB_{16}$

40. Realizar las siguientes restas:

(a) $51_{16} - 40_{16}$ **(b)** $C8_{16} - 3A_{16}$ **(c)** $FD_{16} - 88_{16}$

SECCIÓN 2.9 Números octales

41 Convertir a decimal los siguientes números octales:

(a) 12_8 (b) 27_8 (c) 56_8 (d) 64_8 (e) 103_8

(f) 557_8 (g) 163_8 (h) 1024_8 (i) 7765_8

42. Convertir a octal los siguientes números decimales utilizando la división sucesiva por 8:

(a) 15 **(b)** 27 **(c)** 46 **(d)** 70

(e) 100 **(f)** 142 **(g)** 219 **(h)** 435

43. Convertir a binario los siguientes números octales:

(a) 13_8 **(b)** 57_8 **(c)** 101_8 **(d)** 321_8 **(e)** 540_8

(f) 4653_8 **(g)** 13271_8 **(h)** 45600_8 **(i)** 100213_8

44. Convertir a octal los siguientes números binarios:

(a) 111 **(b)** 10 **(c)** 110111

(d) 101010 **(e)** 1100 **(f)** 1011110

(g) 101100011001 **(h)** 10110000011 **(i)** 111111101111000

SECCIÓN 2.10 Código decimal binario (BCD)

45. Convertir los siguiente números decimales a BCD 8421:

(a) 10 **(b)** 13 **(c)** 18 **(d)** 21 **(e)** 25 **(f)** 36

(g) 44 **(h)** 57 **(i)** 69 **(j)** 98 **(k)** 125 **(l)** 156

46. Convertir los números decimales del Problema 45 a binario normal y comparar el número de bits necesarios con los bits necesarios para BCD.

47. Convertir a BCD los siguientes números decimales:

(a) 104 **(b)** 128 **(c)** 132 **(d)** 150 **(e)** 186

(f) 210 **(g)** 359 **(h)** 547 **(i)** 1051

48. Convertir a decimal los siguientes números BCD:

(a) 0001 **(b)** 0110 **(c)** 1001

(d) 00011000 **(e)** 00011001 **(f)** 00110010

(g) 01000101 **(h)** 10011000 **(i)** 100001110000

49. Convertir a decimal los siguientes números BCD:

(a) 10000000 **(b)** 001000110111

(c) 001101000110 **(d)** 010000100001

(e) 011101010100 **(f)** 100000000000

(g) 100101111000 **(h)** 0001011010000011

(i) 1001000000011000 **(j)** 0110011001100111

50. Sumar los siguientes números BCD:

(a) 0010 + 0001 **(b)** 0101 + 0011

(c) 0111 + 0010 **(d)** 1000 + 0001

(e) 00011000 + 00010001 **(f)** 01100100 + 00110011

(g) 01000000 + 01000111 **(h)** 10000101 + 00010011

51. Sumar los siguientes números BCD:

(a) 1000 + 0110 **(b)** 0111 + 0101
(c) 1001 + 1000 **(d)** 1001 + 0111
(e) 00100101 + 00100111 **(f)** 01010001 + 01011000
(g) 10011000 + 10010111 **(h)** 010101100001 + 011100001000

52. Convertir a BCD cada pareja de números decimales y sumarlos como se indica:

(a) 4 + 3 **(b)** 5 + 2 **(c)** 6 + 4 **(d)** 17 + 12
(e) 28 + 23 **(f)** 65 + 58 **(g)** 113 + 101 **(h)** 295 + 157

SECCIÓN 2.11 Códigos digitales

53. En una determinada aplicación se producen ciclos de una secuencia binaria de 4 bits de 1111 a 0000 de forma periódica. Existen cuatro variaciones de bit, y debido a retrasos del circuito, estas variaciones pueden no producirse en el mismo instante. Por ejemplo, si el LSB cambia el primero, entonces durante la transición de 1111 a 0000 aparecerá el número 1110, y puede ser mal interpretado por el sistema. Ilustrar cómo resuelve este problema el código Gray.

54. Convertir a código Gray los números binarios:

(a) 11011 **(b)** 1001010 **(c)** 1111011101110

55. Convertir a binario los números en código Gray:

(a) 1010 **(b)** 00010 **(c)** 11000010001

56. Convertir a código ASCII cada uno de los siguientes números decimales. Utilice la Tabla 2.7

(a) 1 **(b)** 3 **(c)** 6 **(d)** 10 **(e)** 18 **(f)** 29 **(g)** 56 **(h)** 75 **(i)** 107

57. Determinar el carácter de cada uno de los siguientes códigos ASCII. Utilice la Tabla 2.7.

(a) 0011000 **(b)** 1001010 **(c)** 0111101
(d) 0100011 **(e)** 0111110 **(f)** 1000010

58. Decodificar el siguiente mensaje codificado en ASCII:

1001000 1100101 1101100 1101100 1101111 0101110
0100000 1001000 1101111 1110111 0100000 1100001
1110010 1100101 0100000 1111001 1101111 1110101
0111111

59. Escribir en hexadecimal el mensaje del Problema 58.

60. Convertir a código ASCII la siguiente instrucción de programa para una computadora:

30 INPUT A, B

SECCIÓN 2.12 Códigos de detección y corrección de errores

61. Determinar cuáles de los siguientes códigos con paridad par son erróneos:

(a) 100110010 **(b)** 011101010 **(c)** 10111111010001010

62. Determinar cuáles de los siguientes códigos con paridad impar son erróneos:

(a) 11110110 **(b)** 00110001 **(c)** 01010101010101010

63. Añadir el bit de paridad par apropiado a los siguientes bytes de datos:

(a) 10100100 **(b)** 00001001 **(c)** 11111110

64. Determinar el código Hamming de paridad par para los bits de datos 1100.

65. Determinar el código Hamming de paridad impar para los bits de datos 11001.

66. Corregir cualquier error que pueda haber en los siguientes códigos Hamming con paridad par.

(a) 1110100

(b) 1000111

67. Corregir cualquier error que pueda haber en los siguientes códigos Hamming con paridad impar.

(a) 110100011

(b) 100001101

RESPUESTAS

REVISIONES DE CADA SECCIÓN

SECCIÓN 2.1 **Números**

1. **(a)** 1370: 10 **(b)** 6725: 100 **(c)** 7051: 1000 **(d)** 58,72: 0.1

2. **(a)** 51 = (5 x10) + (1 × 1) **(b)** 137 = (1 × 100) + (3 × 10) + (7 × 1)

(c) 1492 = (1 × 1000) + (4 x100) + (9 × 10) + (2 × 1)

(d) 106,58 = (1 × 100) + (0 × 10) + (6 × 1) + (5 × 0,1) + (8 × 0,01)

SECCIÓN 2.2 **Números binarios**

1. $2^8 - 1 = 255$

2. El peso de 16.

3. 10111101,011 = 189,375

SECCIÓN 2.3 **Conversión decimal-binario**

1. **(a)** 23 = 10111 **(b)** 57 =111001 **(c)** 45,5 = 101101,1

2. **(a)** 14 = 1110 **(b)** 21 = 10101 **(c)** 0,375 = 0,011

SECCIÓN2.4 **Aritmética binaria**

1. **(a)** 1101 + 1010 = 10111 **(b)** 10111 + 01101 = 100100

2. **(a)** 1101 − 0100 = 1001 **(b)** 1001 − 0111 = 0010

3. **(a)** 110 × 111 = 101010 **(b)** 1100 ÷ 011 = 100

SECCIÓN 2.5 **Complemento a 1 y complemento a 2 de los números binarios**

1. **(a)** Complemento a 1 de 00011010 = 11100101

(b) Complemento a 1 de 11110111 = 00001000

(c) Complemento a 1 de 10001101 = 01110010

2. **(a)** Complemento a 2 de 00010110 = 11101010

(b) Complemento a 2 de 11111100 = 00000100

(c) Complemento a 2 de 10010001 = 01101111

SECCIÓN 2.6 **Números con signo**

1. Signo-magnitud: +9 = 00001001

2. Complemento a 1: −33 = 11011110

3. Complemento a 2: −46 = 11010010

4. Bit de signo, exponente y mantisa

SECCIÓN 2.7 Operaciones aritméticas de números con signo

1. Casos de suma: el número positivo es el mayor, el número negativo es el mayor (en valor absoluto), ambos son positivos y ambos son negativos.
2. 00100001 + 10111100 = 11011101
3. 01110111 − 00110010 = 01000101
4. El signo del producto es positivo.
5. 00000101 × 01111111 = 01001111011
6. El signo del cociente es negativo.
7. 00110000 ÷ 00001100 = 00000100

SECCIÓN 2.8 Números hexadecimales

1. **(a)** 10110011 = $B3_{16}$ **(b)** 110011101000 = $CE8_{16}$
2. **(a)** 57_{16} = 01010111 **(b)** $3A5_{16}$ = 001110100101 **(c)** $F8OB_{16}$ = 1111100000001011
3. $9B30_{16}$ = $39{,}728_{10}$
4. $573_{10} = 23D_{16}$
5. **(a)** $18_{16} + 34_{16} = 4C_{16}$ **(b)** $3F_{16} + 2A_{16} = 69_{16}$
6. **(a)** $75_{16} - 21_{16} = 54_{16}$ **(b)** $94_{16} - 5C_{16} = 38_{16}$

SECCIÓN 2.9 Números octales

1. **(a)** $73_8 = 59_{10}$ **(b)** $125_8 = 85_{10}$
2. **(a)** $98_{10} = 142_8$ **(b)** $163_{10} = 243_8$
3. **(a)** 46_8 = 100110 **(b)** 723_8 = 111010011 **(c)** 5624_8 = 101110010100
4. **(a)** 110101111 = 657_8 **(b)** 1001100010 = 1142_8 **(c)** 10111111001 = 2771_8

SECCIÓN 2.10 Código decimal binario (BCD)

1. **(a)** 0010: 2 **(b)** 1000: 8 **(c)** 0001: 1 **(d)** 0100: 4
2. **(a)** 6_{10} = 0110 **(b)** 15_{10} = 00010101 **(c)** 273_{10} = 001001110011
   (d) 849_{10} = 100001001001
3. **(a)** 10001001 = 89_{10} **(b)** 001001111000 = 278_{10} **(c)** 000101010111 = 157_{10}
4. Una suma de 4 bits no es válida cuando es mayor que 9_{10}.

SECCIÓN 2.11 Códigos digitales

1. **(a)** 1100_2 = 1010 Gray **(b)** 1010_2 = 1111 Gray **(c)** 11010_2 = 10111 Gray
2. **(a)** 1000 Gray = 1111_2 **(b)** 1010 Gray = 1100_2 **(c)** 11101 Gray = 10110_2
3. **(a)** K: 1001011 → $4B_{16}$ **(b)** r: 1110010 → 72_{16}
   (c) $: 0100100 → 24_{16} **(d)** +: 0101011 → $2B_{16}$

SECCIÓN 2.12 Códigos de detección y corrección de errores

1. **(c)** 0101 tiene un error.
2. **(d)** 11111011 tiene un error.
3. **(a)** 1010100**1** **(b)** 0100000**1** **(c)** 1110111**0** **(d)** 1000110**1**
4. Cuatro bits de paridad
5. **1** **0** 0 **0** 0 1 1 (bits de paridad en negrita)

PROBLEMAS RELACIONADOS

2.1 9 tiene un valor de 900, 3 tiene un valor de 30, 9 tiene un valor de 9.

2.2 6 tiene un valor de 60, 7 tiene un valor de 7, 9 tiene un valor de 9/10 (0,9), 2 tiene un valor de 2/100 (0,02), 4 tiene un valor de 4/1000 (0,004).

2.3 10010001 = 128 + 16 + 1 = 145 **2.4** 10.111 = 2 + 0.5 + 0.25 + 0.125 = 2.875

2.5 125 = 64 + 32 + 16 + 8 + 4 + 1 = 1111101 **2.6** 39 = 100111

2.7 1111 + 1100 = 11011 **2.8** 111 − 100 = 011 **2.9** 110 − 101 = 001

2.10 1101 × 1010 = 10000010 **2.11** 1100 ÷ 100 = 11 **2.12** 00110101

2.13 01000000 **2.14** Véase la Tabla 2.16. **2.15** 01110111 = $+119_{10}$

	Signo-magnitud	Complemento a 1	Complemento a 2
+19	00010011	00010011	00010011
−19	10010011	11101100	11101101

Tabla 2.16

2.16 11101011 = -20_{10} **2.17** 11010111 = -41_{10}

2.18 11000010001010011000000000 **2.19** 01010101 **2.20** 00010001

2.21 1001000110 **2.22** (83)(−59) = −4897 (10110011011111 en complemento a 2)

2.23 100 ÷ 25 = 4 (0100) **2.24** $4F79C_{16}$ **2.25** 0110101111010011_2

2.26 $6BD_{16} = 011010111101 = 2^{10} + 2^9 + 2^7 + 2^5 + 2^4 + 2^3 + 2^2 + 2^0$

$= 1024 + 512 + 128 + 32 + 16 + 8 + 4 + 1 = 1725_{10}$

2.27 $60A_{16} = (6 \times 256) + (0 \times 16) + (10 \times 1) = 1546_{10}$

2.28 $2591_{10} = A1F_{16}$ **2.29** $4C_{16} + 3A_{16} = 86_{16}$

2.30 $BCD_{16} - 173_{16} = A5A_{16}$

2.31 **(a)** $001011_2 = 11_{10} = 13_8$ **(b)** $010101_2 = 21_{10} = 25_8$

(c) $001100000_2 = 96_{10} = 140_8$ **(d)** $111101010110_2 = 3926_{10} = 7526_8$

2.32 1250762_8 **2.33** 1001011001110011 **2.34** $82{,}276_{10}$

2.35 1001100101101000 **2.36** 10000010

2.37 **(a)** 111011 (Gray) **(b)** 111010_2

2.38 La secuencia de códigos para 80 INPUT Y es $38_{16}30_{16}20_{16}49_{16}4E_{16}50_{16}55_{16}54_{16}20_{16}59_{16}$

2.39 01001011 **2.40** Sí **2.41** *1*1*1*0000 **2.42** *00*1*0*100*0*1

2.43 El bit en la posición 010 (2) es erróneo. Código corregido: 0011001.

2.44 El bit en la posición 0010 (2) es erróneo. Código corregido: 111111000.

AUTOTEST

1. (d) **2.** (a) **3.** (b) **4.** (c) **5.** (c) **6.** (a) **7.** (d) **8.** (b)

9. (d) **10.** (a) **11.** (c) **12.** (d) **13.** (d) **14.** (b) **15.** (c) **16.** (a)

17. (c) **18.** (a) **19.** (b)

3

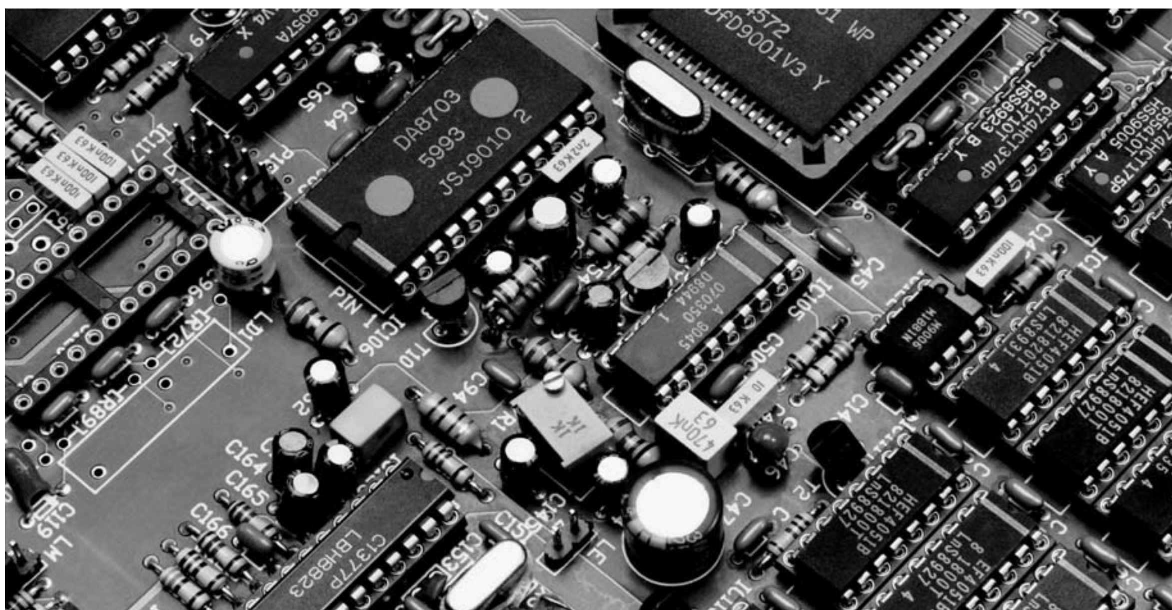
PUERTAS LÓGICAS

CONTENIDO DEL CAPÍTULO

3.1 El inversor
3.2 La puerta AND
3.3 La puerta OR
3.4 La puerta NAND
3.5 La puerta NOR
3.6 Puertas OR–exclusiva y NOR–exclusiva
3.7 Lógica programable
3.8 Lógica de función fija
3.9 Localización de averías

OBJETIVOS DEL CAPÍTULO

- Describir el funcionamiento del inversor y de las puertas AND y OR.
- Describir el funcionamiento de las puertas NAND y NOR.
- Expresar las operaciones de las puertas NOT, AND, OR, NAND y NOR mediante el álgebra de Boole.
- Describir el funcionamiento de las puertas OR–exclusiva y NOR–exclusiva.
- Reconocer y utilizar los símbolos distintivos y los símbolos rectangulares de las puertas lógicas según el estándar ANSI/IEEE 91–1984.



- Elaborar los diagramas de tiempos que muestran las relaciones de tiempo de las entradas y las salidas de las diferentes puertas lógicas.
- Establecer las comparaciones básicas entre las principales tecnologías de circuitos integrados: TTL y CMOS.
- Explicar las diferencias entre las series de las familias TTL y CMOS.
- Definir, para las puertas lógicas, los siguientes parámetros: *tiempo de retardo de propagación, disipación de potencia, producto velocidad–potencia y fan–out*.
- Enumerar circuitos integrados de función fija que contengan varias puertas lógicas.
- Utilizar cada puerta lógica en aplicaciones sencillas.
- Localización de averías en las puertas lógicas debidas a circuitos abiertos o cortocircuitos, utilizando el pulsador y la sonda lógica o el osciloscopio.

PALABRAS CLAVE

- Inversor
- Tabla de verdad
- Diagrama de tiempos
- Álgebra booleana
- Complemento
- Puerta AND
- Habilitar
- Puerta OR
- Puerta NAND
- Puerta NOR
- Puerta OR–exclusiva
- Puerta NOR–exclusiva
- Matriz AND
- Fusible
- Antifusible
- EPROM
- EEPROM
- SRAM
- Dispositivo objetivo
- JTAG
- CMOS
- TTL
- Tiempo de retardo de propagación
- Fan–out
- Carga unidad

INTRODUCCIÓN

Este capítulo hace énfasis en el funcionamiento lógico, las aplicaciones y la localización de averías de las puertas lógicas. Se cubre la relación entre las formas de onda de entrada y de salida de una puerta utilizando los diagramas de tiempos.

Los símbolos lógicos que se usan para representar las puertas lógicas están de acuerdo con el estándar ANSI/IEEE 91–1984. Este estándar ha sido adoptado por la industria privada, y la industria militar lo utiliza para su documentación interna así como para sus publicaciones.

En este capítulo se aborda tanto la lógica programable como la lógica de función fija. Puesto que en todas las aplicaciones se usan los circuitos integrados (CI), generalmente, la función lógica de un dispositivo es más importante para el técnico que los detalles de operación del circuito en el nivel de componentes en el interior del CI. Por tanto, la cobertura detallada de los dispositivos en el nivel de componente puede tratarse como un tema opcional. Para aquéllos que lo necesiten y tengan tiempo, en el Capítulo 14 se cubren las tecnologías de los circuitos integrados digitales, haciéndose referencia a partes del mismo a lo largo del texto. *Sugerencia*: repase la Sección 1.3 antes de comenzar con este capítulo.

DISPOSITIVOS LÓGICOS DE FUNCIÓN FIJA

(SERIES CMOS Y TTL)

74XX00	74XX02	74XX04
74XX08	74XX10	74XX11
74XX20	74XX21	74XX27
74XX30	74XX32	74XX86
74XX266		

3.1 EL INVERSOR

El inversor (circuito NOT) realiza la operación denominada *inversión* o *complementación*. El inversor cambia un nivel lógico al nivel opuesto. En términos de bits, cambia un 1 por un 0, y un 0 por 1.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Identificar los indicadores de negación y polaridad. ■ Identificar un inversor tanto mediante su símbolo distintivo como por su símbolo rectangular. ■ Elaborar la tabla de verdad del inversor. ■ Describir el funcionamiento lógico de un inversor.

En la Figura 3.1 se muestran los símbolos lógicos estándar del ***inversor***. La parte (a) muestra los *símbolos distintivos*, y la (b) muestra los *símbolos rectangulares*. En este texto se usan los símbolos distintivos; sin embargo, los símbolos rectangulares suelen encontrarse en las documentaciones industriales, por lo que debería familiarizarse con ellos. Los símbolos lógicos cumplen el estándar **ANSI/IEEE** 91–1984.

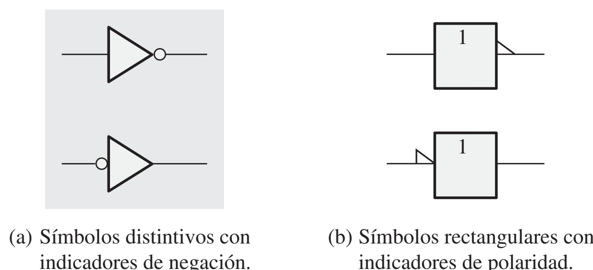


Figura 3.1 Símbolos lógicos estándar de la puerta inversora (Estándar ANSI/IEEE 91–1984).

Los indicadores de negación y de polaridad

El indicador de negación es un "círculo" (○) que indica **inversión** o *complementación*, cuando aparece en la entrada o en la salida de un elemento lógico, tal como muestra la Figura 3.1(a) para el inversor. Generalmente, las entradas se sitúan a la izquierda del símbolo lógico, y la salida a la derecha. Cuando en la entrada hay un círculo, quiere decir que el estado activo o *verdadero* de la entrada es 0, y se dice que la entrada es activa a nivel BAJO. Cuando el círculo se sitúa en la salida significa que el estado activo o verdadero de salida es 0, y se dice que la salida es activa a nivel BAJO. La ausencia de círculo en la entrada o en la salida significa que el estado activo o verdadero es 1 y, en este caso, se dice que la entrada o la salida es activa a nivel ALTO.

El indicador de polaridad o de nivel es un "triángulo" (◺) que indica inversión cuando aparece a la entrada o a la salida de un elemento lógico, como muestra la Figura 3.1(b). Cuando se presenta a la entrada, significa que un nivel BAJO es el estado de entrada activo o verdadero. Cuando se presenta a la salida, significa que un nivel BAJO es el estado de salida activo o verdadero.

Ambos indicadores (círculo y triángulo) pueden utilizarse tanto en los símbolos distintivos como en los rectangulares. La Figura 3.1(a) indica el principal uso de los símbolos del inversor en este texto. Observe que un cambio de posición del indicador de polaridad o de negación no implica un cambio en el modo de funcionamiento del inversor.

Tabla de verdad del inversor

Cuando se aplica un nivel ALTO a la entrada de un inversor, en su salida se presenta un nivel BAJO. Cuando se aplica un nivel BAJO a la entrada, en su salida se presenta un nivel ALTO. En la Tabla 3.1 se resume esta

operación. Esta tabla muestra la salida para cada posible entrada en términos de niveles y bits correspondientes. Una tabla tal como ésta se llama **tabla de verdad**.

Entrada	Salida
BAJO (0)	ALTO (1)
ALTO (1)	BAJO (0)

Tabla 3.1 Tabla de verdad del inversor.

Funcionamiento del inversor

La Figura 3.2 muestra la salida de un inversor para un impulso de entrada, donde t_1 y t_2 indican los puntos que corresponden a los impulsos de entrada y salida.

Cuando la entrada está a nivel BAJO, la salida está a nivel ALTO; cuando la entrada está a nivel ALTO, la salida está a nivel BAJO, lo que da lugar a un impulso de salida invertido.

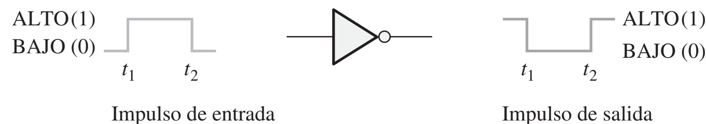


Figura 3.2 Funcionamiento del inversor con un impulso de entrada.

Diagramas de tiempos

▲ *Un diagrama de tiempos muestra cómo se relacionan dos o más señales en el tiempo.*

Recuerde del Capítulo 1 que un **diagrama de tiempos o cronograma** es básicamente *una gráfica que presenta de forma precisa las relaciones de dos o más formas de onda en función del tiempo*. Por ejemplo, la relación de tiempo del impulso de salida respecto al impulso de entrada de la Figura 3.2 puede representarse con un sencillo diagrama de tiempos, alineando los dos impulsos de modo que las ocurrencias de los flancos se presenten en los instantes de tiempo correctos. El flanco de subida del impulso de entrada y el flanco de bajada del impulso de salida se producen al mismo tiempo (idealmente). Igualmente, el flanco de bajada del impulso de entrada y el flanco de subida del impulso de salida se producen al mismo tiempo (idealmente). En la Figura 3.3 se muestra la relación de tiempos. Los diagramas de tiempos son muy útiles para ilustrar las relaciones de las señales digitales de impulsos múltiples.

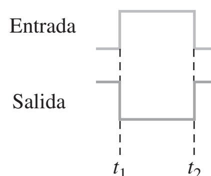


FIGURA 3.3 Diagrama de tiempos para el caso de la Figura 3.2.

Expresión lógica del inversor

En el **álgebra booleana**, que son las matemáticas de los circuitos lógicos y que se cubrirán en el Capítulo 4, una variable se designa mediante una letra. El **complemento** de una variable se designa mediante una barra

EJEMPLO 3.1

Al inversor de la Figura 3.4 se le aplica una señal. Determinar la forma de onda de salida correspondiente a la entrada y dibujar el diagrama de tiempos. De acuerdo con el emplazamiento del círculo ¿cuál es el estado activo de salida?



FIGURA 3.4

Solución La forma de onda de salida es exactamente la opuesta a la de entrada (es la entrada invertida), como se muestra en la Figura 3.5, que es el cronograma básico. El estado activo o verdadero de salida es 0.

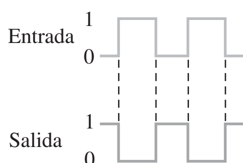


FIGURA 3.5

Problema relacionado* Si el inversor tiene el indicador negativo (círculo) en la entrada en lugar de en la salida, ¿cómo afecta esto al diagrama de tiempos?

* Las respuestas se encuentran al final del capítulo.

▲ *El álgebra booleana utiliza variables y operadores para describir un circuito lógico.*

encima de la letra. Una variable puede tomar uno de dos valores, 1 ó 0. Si una variable dada es 1, su complemento es 0, y viceversa.

El modo de operación de un inversor (circuito NOT) puede expresarse del siguiente modo: si la variable de entrada se designa por A y la variable de salida por X, entonces

$$X = \overline{A}$$

Esta expresión establece que la salida es el complemento de la entrada, de modo que si $A = 0$, entonces $X = 1$, y si $A = 1$, entonces $X = 0$. La Figura 3.6 ilustra esto. La variable complementada $\overline{A}$ se lee "A negada" o "A complementada".

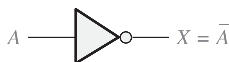


FIGURA 3.6 El inversor complementa una variable de entrada.

Aplicación

La Figura 3.7 muestra un circuito que genera el complemento a 1 de un número binario de 8 bits. Los bits del número binario se aplican a las entradas del inversor y el complemento a 1 se obtiene en las salidas.

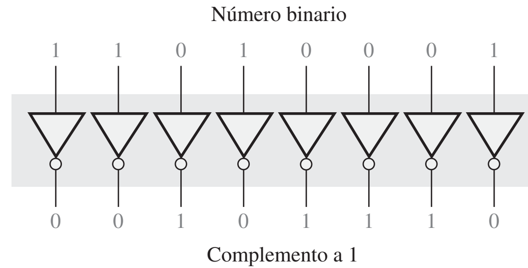


FIGURA 3.7 Ejemplo de un circuito que genera el complemento a 1 utilizando inversores.

REVISIÓN DE LA SECCIÓN 3.1

Las respuestas se encuentran al final del capítulo.

1. Cuando en la entrada de un inversor hay un 1, ¿cuál es la salida?
2. En la entrada de un inversor se requiere un impulso activo a nivel ALTO (el nivel ALTO es verdadero, y el nivel BAJO no).
   (a) Dibujar el símbolo lógico correspondiente, utilizando el indicador de negación y el símbolo distintivo para el inversor de esta aplicación.
   (b) Describir la salida cuando un impulso de subida se aplica a la entrada del inversor.

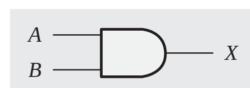
3.2 LA PUERTA AND

La puerta AND es una de las puertas básicas con la que se construyen todas las funciones lógicas. Una puerta AND puede tener dos o más entradas y realiza la operación que se conoce como multiplicación lógica.

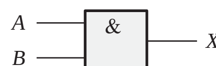
Al finalizar este capítulo, el lector deberá ser capaz de:

■ Identificar una puerta AND mediante su símbolo distintivo y su símbolo rectangular. ■ Describir la operación lógica de una puerta AND. ■ Generar la tabla de verdad de una puerta AND con cualquier número de entradas. ■ Generar el cronograma de una puerta AND para cualquier forma de onda especificada en sus entradas. ■ Escribir la expresión lógica de una puerta AND con cualquier número de entradas. ■ Analizar ejemplos de aplicaciones de la puerta AND.

El término *puerta* se usa para describir un circuito que realiza una operación lógica básica. La puerta AND tiene dos o más entradas y una única salida, como indican los símbolos lógicos estándar mostrados en la Figura 3.8. En cada uno de los símbolos, las entradas se sitúan a la izquierda y la salida a la derecha. Se muestran puertas con dos entradas, pero una puerta AND puede tener cualquier número de entradas superior a éste. Aunque en los ejemplos se utilizan ambos tipos de símbolos, distintivos y rectangulares, en este libro, predominantemente, se emplea el símbolo distintivo de la Figura 3.8(a).



(a) Símbolo distintivo



(b) Símbolo rectangular, identificado mediante el carácter AND (&)

FIGURA 3.8 Símbolos lógicos estándar de la puerta AND con dos entradas (estándar ANSI/IEEE 91–1984).



NOTAS INFORMÁTICAS

Las puertas lógicas son los bloques de construcción de las computadoras. La mayor parte de las funciones en una computadora, con la excepción de ciertos tipos de memoria, se implementan mediante puertas lógicas utilizadas a muy gran escala. Por ejemplo, un microprocesador, que es la parte principal de una computadora, consta de cientos de miles de puertas lógicas.

Funcionamiento de la puerta AND

▲ *Una puerta AND puede tener más de dos entradas.*

La ***puerta AND*** genera una salida a nivel ALTO *sólo* cuando *todas* las entradas están a nivel ALTO. Cuando cualquiera de la entradas está a nivel BAJO, la salida se pone a nivel BAJO. Por tanto, el propósito básico de una puerta AND es determinar cuándo ciertas condiciones de entrada son simultáneamente verdaderas, como indican todas sus entradas estando a nivel ALTO, y producir una salida a nivel ALTO, para indicar que esas condiciones son verdaderas. Las entradas de la puerta AND de dos entradas de la Figura 3.8 se designan mediante A y B, y la salida con X, luego podemos establecer que el funcionamiento de la puerta es el siguiente:

En una puerta AND de dos entradas, la salida *X* es un nivel ALTO si *A* y *B* están a nivel ALTO; y *X* es un nivel BAJO si *A* es un nivel BAJO, o si *B* es un nivel BAJO, o si *A* y *B* están a nivel BAJO.

La Figura 3.9 ilustra una puerta AND de 2 entradas en la que se indican las cuatro posibles combinaciones de entrada y el resultado correspondiente a cada una de ellas.

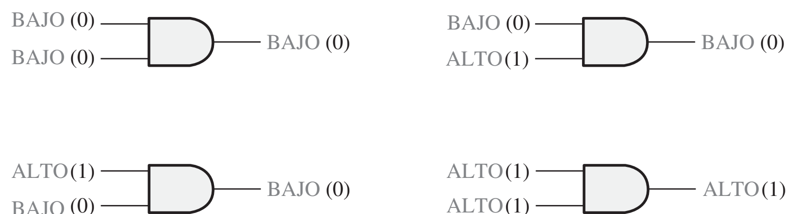


FIGURA 3.9 Todos los posibles niveles lógicos para una puerta AND de dos entradas.

Tabla de verdad de la puerta AND

▲ *En una puerta AND, si todas las entradas están a nivel ALTO, la salida es un nivel ALTO.*

La operación lógica de una puerta puede expresarse mediante una tabla de verdad, en la que se enumeran todas las combinaciones de entrada con las correspondientes salidas, como muestra la Tabla 3.2 para una puerta AND de dos entradas. La tabla de verdad puede ampliarse para cualquier número de entradas. Aunque los términos nivel ALTO y nivel BAJO dan un sentido "físico" a los estados de entrada y salida, la tabla de verdad se presenta con 1s y 0s, ya que un nivel ALTO es equivalente a un 1, y un nivel BAJO es equivalente a 0 en lógica positiva. Para cualquier puerta AND, independientemente del número de entradas, la salida es un nivel ALTO *sólo* cuando *todas* las entradas están a nivel ALTO.

El número total de posibles combinaciones de entradas binarias a una puerta viene determinado por la siguiente fórmula:

Ecuación 3.1 $$N = 2^n$$

donde N es el número de posibles combinaciones de entrada y n es el número de variables de entrada:

Entradas		Salida
A	*B*	*X*
0	0	0
0	1	0
1	0	0
1	1	1
1 = ALTO, 0 = BAJO		

TABLA 3.2 Tabla de verdad de una puerta AND de dos entradas.

Para dos variables de entrada: $N = 2^2 = 4$ combinaciones
Para tres variables de entrada: $N = 2^3 = 8$ combinaciones
Para cuatro variables de entrada: $N = 2^4 = 16$ combinaciones

Utilizando la Ecuación 3.1 se puede determinar el número de combinaciones de bits de entrada para puertas con cualquier número de entradas.

EJEMPLO 3.2

(a) Desarrollar la tabla de verdad de una puerta AND de 3 entradas.
(b) Determinar el número total de posibles combinaciones de entrada para una puerta AND de 4 entradas.

Solución

(a) Para una puerta AND de 3 entradas existen ocho posibles combinaciones de entrada ($2^3 = 8$). Las entradas de la tabla de verdad (Tabla 3.3) muestran las ocho posibles combinaciones de tres bits. La salida es siempre 0, excepto cuando los tres bits de entrada son 1.

Entradas			Salidas
A	*B*	*C*	*X*
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

TABLA 3.3

(b) $N = 2^4 =$ **16**. Para una puerta AND de 4 entradas existen 16 posibles combinaciones de bits de entrada.

Problema relacionado Desarrollar la tabla de verdad para una puerta AND de 4 entradas.

Funcionamiento con trenes de impulsos

En la mayoría de las aplicaciones, las entradas a una puerta no son niveles estacionarios sino tensiones que cambian frecuentemente entre los niveles lógicos ALTO y BAJO. Ahora vamos a ver el funcionamiento de las puertas AND con entradas que son señales digitales, teniendo en mente que una puerta AND obedece a su tabla de verdad, independientemente de que sus entradas sean niveles constantes o señales que varíen de un nivel a otro.

Al examinar el funcionamiento de una puerta AND con trenes de impulsos, nos fijaremos en los niveles de entrada para determinar el nivel de salida en cualquier instante dado. En la Figura 3.10, ambas entradas *A* y *B* están a nivel ALTO (1) durante el intervalo de tiempo t_1, por lo que la salida *X* en este intervalo estará a nivel ALTO (1). Durante el intervalo t_2, la entrada *A* está a nivel BAJO (0) y la entrada *B* está a nivel ALTO (1), por lo que la salida se pondrá a nivel BAJO (0). De nuevo, durante el intervalo t_3, ambas entradas están a nivel ALTO (1) y, por tanto, la salida está a nivel ALTO (1). Durante el intervalo t_4, la entrada *A* está a nivel ALTO (1) y la *B* está a nivel BAJO (0), luego la salida está a nivel BAJO (0). Por último, durante el intervalo t_5, la entrada *A* está a nivel BAJO (0) y la entrada *B* está a nivel BAJO (0) y, por tanto, la salida está a nivel BAJO (0). Como ya sabe, un diagrama de las señales de entrada y de salida en función del tiempo se llama *diagrama de tiempos o cronograma*.

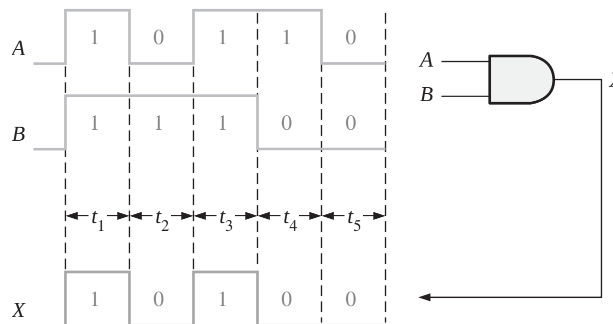


Figura 3.10 Ejemplo de funcionamiento de una puerta AND con trenes de impulsos, y cronograma que muestra las relaciones entre las entradas y la salida.

EJEMPLO 3.3

Si se aplican las formas de onda *A* y *B* de la Figura 3.11 a las entradas de una puerta AND, ¿cuál es la forma de onda resultante de salida?

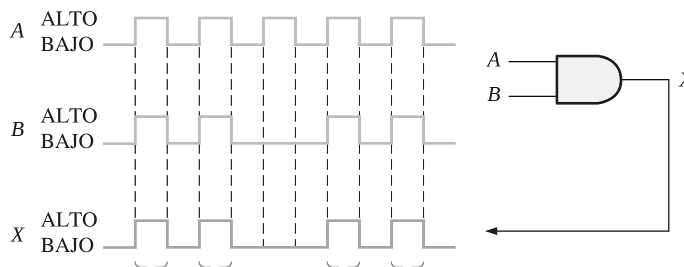


FIGURA 3.11

Solución	La forma de onda de salida X sólo está a nivel ALTO cuando A y B están a nivel ALTO, tal y como se muestra en el diagrama de tiempos de la Figura 3.11.
Problema relacionado	Determinar la forma de onda de salida y dibujar el diagrama de tiempos si los impulsos segundo y cuarto de la señal A de la Figura 3.11 se reemplazan ambos por un nivel BAJO.

Es importante recordar que cuando se analiza el funcionamiento con trenes de impulsos de las puertas lógicas, hay que poner especial cuidado en las relaciones de tiempo de todas las entradas entre sí y con la salida.

EJEMPLO 3.4

Para las dos formas de onda de entrada, *A* y *B*, de la Figura 3.12, dibujar la onda de salida mostrando su relación con las entradas.

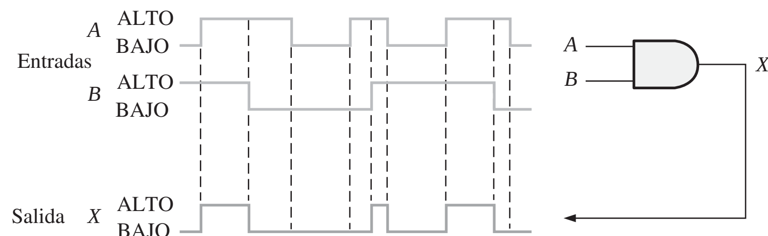


FIGURA 3.12

Solución La onda de salida está a nivel ALTO sólo cuando ambas entradas están a nivel ALTO, como se muestra en el diagrama de tiempos.

Problema relacionado Obtener la onda de salida si la entrada *B* de la puerta AND de la Figura 3.12 siempre está a nivel ALTO.

EJEMPLO 3.5

Para la puerta AND de 3 entradas de la Figura 3.13, determinar la forma de onda de salida con respecto a las entradas.

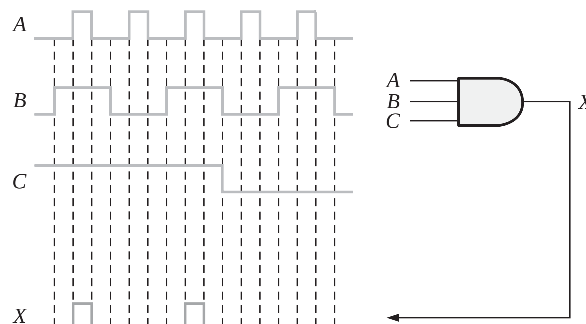


FIGURA 3.13

Solución	La onda de salida X de una puerta AND de 3 entradas está a nivel ALTO sólo cuando las tres entradas, *A*, *B* y *C* están a nivel ALTO.
Problema relacionado	¿Cuál es la onda de salida de la puerta AND de la Figura 3.13 si la entrada *C* está siempre a nivel ALTO?

Expresiones lógicas para la puerta AND

La función lógica AND de dos variables se representa matemáticamente colocando un punto entre las dos variables, $A \cdot B$, o simplemente escribiendo las letras juntas sin el punto, *AB*. Normalmente, utilizaremos esta última notación, ya que es más cómoda de escribir.

La **multiplicación booleana** sigue las mismas reglas básicas que gobiernan la multiplicación binaria, que hemos tratado en el Capítulo 2 y son las siguientes:

$0 \cdot 0 = 0$
$0 \cdot 1 = 0$
$1 \cdot 0 = 0$
$1 \cdot 1 = 1$

La multiplicación booleana es lo mismo que la función AND.



NOTAS INFORMÁTICAS

Las computadoras pueden utilizar todas las operaciones lógicas básicas cuando tienen que manipular de forma selectiva ciertos bits pertenecientes a uno o más bytes de datos. La manipulación selectiva de bits se lleva a cabo mediante una *máscara*. Por ejemplo, para borrar (poner a 0) los cuatro bits de la derecha de un byte de datos manteniendo los cuatro bits de la izquierda, aplique la operación AND al byte de datos y al valor 11110000. Observe que cualquier bit que se opere (AND) con cero dará 0 y cualquier byte al que se aplique la operación AND con el valor 1 quedará como está. Si se aplica la operación AND al valor 10101010 con la máscara 11110000, el resultado es 10100000.

El funcionamiento de una puerta AND de dos entradas puede expresarse en forma de ecuación como sigue: si una variable de entrada es *A* y la otra variable es *B*, y la variable de salida es *X*, entonces la expresión booleana es

$$X = AB$$

La Figura 3.14(a) muestra la puerta con las variables de entrada y de salida indicadas.

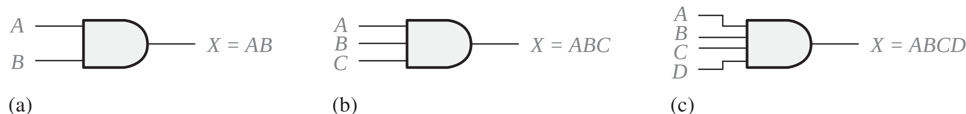


Figura 3.14 Expresión booleana para puertas AND con dos, tres y cuatro entradas.

▲ *Cuando las variables se escriben como en ABC, quiere decir que se aplica la operación AND.*

Para extender la expresión AND a más de dos variables de entrada, simplemente utilice una nueva letra para cada variable de entrada. Por ejemplo, la función de una puerta AND de 3 entradas, se puede expresar así: $X = ABC$, donde *A*, *B* y *C* son las variables de entrada. La expresión para una puerta AND de 4 entradas será $X = ABCD$, y así sucesivamente. Las partes (b) y (c) de la Figura 3.14 muestran, respectivamente, puertas AND de tres y cuatro variables de entrada.

Se puede evaluar el funcionamiento de una puerta AND utilizando las expresiones booleanas que facilitan la salida. Por ejemplo, cada variable de entrada puede ser 1 ó 0, luego para evaluar la puerta AND de dos entradas, se sustituyen dichos valores en la ecuación de salida, $X = AB$, como se muestra en la Tabla 3.4. Esta evaluación muestra que la salida X de una puerta AND es 1 (nivel ALTO) sólo cuando ambas entradas son 1 (nivel ALTO). Se puede hacer un análisis similar para cualquier número de variables de entrada.

A	B	$AB = X$
0	0	$0 \cdot 0 = 0$
0	1	$0 \cdot 1 = 0$
1	0	$1 \cdot 0 = 0$
1	1	$1 \cdot 1 = 1$

TABLA 3.4

Aplicaciones

La puerta AND como un dispositivo de habilitación/inhibición. Una aplicación común de la puerta AND es ***habilitar*** (*enable*) o permitir el paso de una señal (tren de impulsos) de un punto a otro en determinados instantes, e inhibir o impedir el paso en otros instantes.

Un ejemplo sencillo de este particular uso de la puerta AND se muestra en la Figura 3.15, donde la puerta AND controla el paso de una señal (A) a un contador digital. El propósito de este circuito es medir la frecuencia de la señal A. El impulso de habilitación tiene un ancho de exactamente 1 s. Cuando la entrada de habilitación está a nivel ALTO, la señal A pasa a través de la puerta hasta el contador, y cuando la entrada de habilitación está a nivel BAJO, se impide el paso de la señal a través de la puerta.

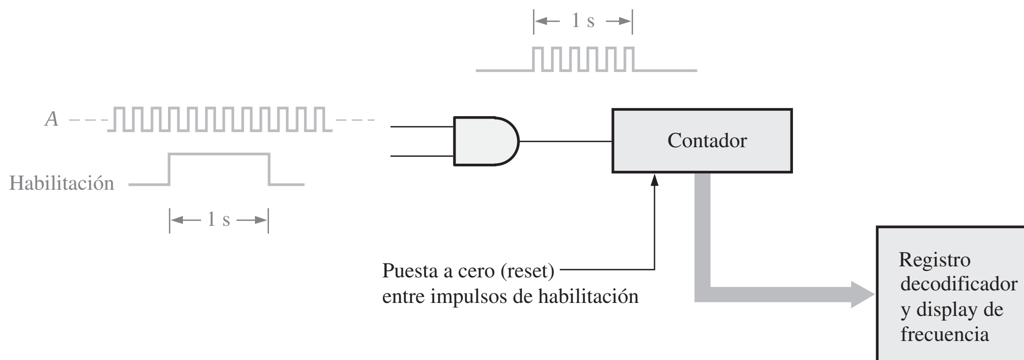


FIGURA 3.15 Una puerta AND que realiza la función de habilitación/inhibición para un contador de frecuencia.

Durante el intervalo de habilitación de 1 segundo, un cierto número de impulsos de la señal A pasan a través de la puerta AND hasta el contador. El número de impulsos que pasa durante 1 s es igual a la frecuencia de la señal. Por ejemplo, la Figura 3.15 muestra una señal en la que seis impulsos duran un segundo, lo que representa una frecuencia de 6 Hz. Si pasan 1.000 impulsos a través de la puerta en el intervalo de 1 s del impulso de habilitación, serán 1.000 impulsos/segundo, es decir una frecuencia de 1.000 Hz.

El contador cuenta el número de impulsos por segundo y genera una salida binaria que pasa al circuito decodificador y al display, en el que se genera una lectura de la frecuencia. El impulso de habilitación se repite en determinados instantes, en los que se lleva a cabo un nuevo recuento por si la frecuencia ha variado, y

el nuevo valor se presentará en el display. Entre los impulsos de habilitación, el contador se pone a cero para reinicializarse cada vez que se produce un impulso de habilitación. La frecuencia actual se almacena en un registro, de modo que el display no se vea afectado al poner a cero el contador.

***Un sistema de alarma para el cinturón de seguridad*.** En la Figura 3.16, se usa una puerta AND en un sencillo sistema de alarma para el cinturón de seguridad del coche, el cual detecta cuándo el interruptor de arranque se ha activado y (*AND*) el cinturón de seguridad no está abrochado. Si el interruptor de arranque se ha activado, la entrada A de la puerta AND se pone a nivel ALTO. Si el cinturón de seguridad no está correctamente abrochado, la entrada *B* de la puerta AND se pone a nivel ALTO. También cuando el interruptor de arranque se activa, se inicializa un temporizador que pone a nivel ALTO la entrada *C* durante 30 s. Si estas tres condiciones se cumplen, es decir, si el interruptor de arranque está activado y (*AND*) el cinturón de seguridad está desabrochado y (*AND*) el temporizador está corriendo, la salida de la puerta AND se pone a nivel ALTO, y una alarma audible se activa para advertir al conductor.

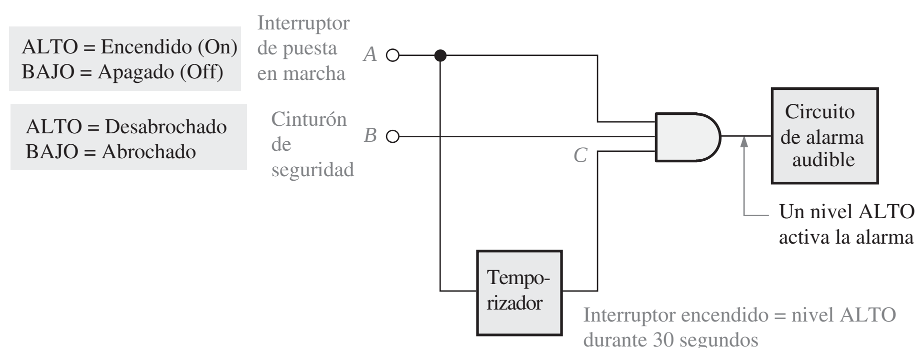


FIGURA 3.16 Un sencillo circuito de alarma para cinturón de seguridad utilizando una puerta AND.

REVISIÓN DE LA SECCIÓN 3.2

1. ¿Cuándo se pone a nivel ALTO la salida de una puerta AND?
2. ¿Cuándo se pone a nivel BAJO la salida de una puerta AND?
3. Describir la tabla de verdad para una puerta AND de cinco entradas.

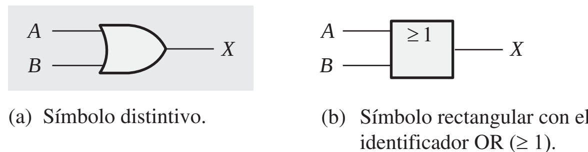
3.3 LA PUERTA OR

La puerta OR es otra de las puertas básicas con las que se construyen todas las funciones lógicas. Una puerta OR puede tener dos o más entradas y realiza la operación que se conoce como suma lógica.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Identificar una puerta OR mediante su símbolo distintivo y su símbolo rectangular. ■ Describir la operación lógica de una puerta OR. ■ Generar la tabla de verdad de una puerta OR con cualquier número de entradas. ■ Generar el diagrama de tiempos de una puerta OR para cualquier forma de onda especificada en sus entradas. ■ Escribir la expresión lógica de una puerta OR con cualquier número de entradas. ■ Desarrollar ejemplos de aplicaciones de la puerta OR.

Una ***puerta OR*** tiene dos o más entradas y una salida, como indican los símbolos lógicos estándar de la Figura 3.17, en la que se muestran puertas OR con dos entradas. Una puerta OR puede tener cualquier número de entradas mayor o igual que dos. Aunque se presentan ambos tipos de símbolos, distintivo y rectangular, en este texto se utilizará el símbolo distintivo de la puerta OR.



(a) Símbolo distintivo.

(b) Símbolo rectangular con el identificador OR (≥ 1).

FIGURA 3.17 Símbolos lógicos estándar de la puerta OR con dos entradas (Estándar ANSI/IEEE 91–1984).

Funcionamiento de la puerta OR

▲ *Una puerta OR puede tener más de dos entradas.*

Una puerta OR genera un nivel ALTO a la salida cuando *cualquiera* de sus entradas está a nivel ALTO. La salida se pone a nivel BAJO sólo cuando todas las entradas están a nivel BAJO. Por tanto, el propósito de una puerta OR es determinar cuándo una o más de sus entradas están a nivel ALTO y generar una salida a nivel ALTO que indique esta condición. Las entradas de la puerta OR de dos entradas de la Figura 3.17 están etiquetadas como *A* y *B*, y la salida como *X*. Podemos establecer el funcionamiento de la puerta como sigue:

En una puerta OR, la salida *X* es un nivel ALTO si cualquiera de las entradas, *A* o *B*, o ambas, están a nivel ALTO; *X* es un nivel BAJO si ambas entradas, *A* y *B*, están a nivel BAJO.

El nivel ALTO es el nivel de salida activo o verdadero para la puerta OR. La Figura 3.18 ilustra la operación lógica para una puerta OR de dos entradas, indicando las cuatro posibles combinaciones de entrada.

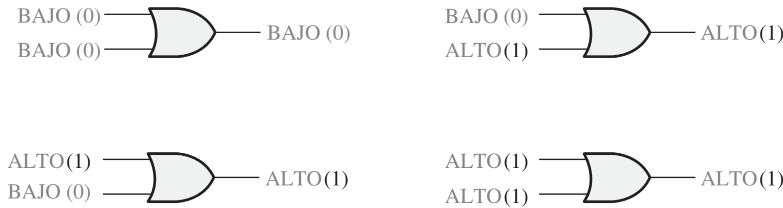


FIGURA 3.18 Todos los posibles niveles lógicos para una puerta OR de 2 entradas.

Tabla de verdad de una puerta OR

▲ *En una puerta OR, si una entrada está a nivel ALTO la salida es un nivel ALTO.*

En la Tabla 3.5 se describe el funcionamiento lógico de una puerta OR de dos entradas. Esta tabla de verdad puede extenderse a cualquier número de entradas e, independientemente del número de entradas, la salida es un nivel ALTO cuando una o más entradas están a nivel ALTO.

Entradas		**Salida**
A	***B***	***X***
0	0	0
0	1	1
1	0	1
1	1	1
1 = ALTO, 0 = BAJO		

TABLA 3.5 Tabla de verdad para una puerta OR de dos entradas.

Funcionamiento con trenes de impulsos

Ahora vamos a ver el funcionamiento de una puerta OR con trenes de impulsos como entradas, teniendo en mente su modo de operación lógico. De nuevo, lo importante en el análisis del funcionamiento de la puerta con trenes de impulsos en las entradas es la relación de tiempos de todas las señales implicadas. Por ejemplo, en la Figura 3.19, las entradas *A* y *B* están a nivel ALTO (1) durante el intervalo t_1, haciendo que la salida *X* esté a nivel ALTO (1). Durante el intervalo t_2, la entrada *A* está a nivel BAJO (0) pero, puesto que la entrada *B* está a nivel ALTO (1), la salida estará a nivel ALTO (1). Durante el intervalo t_3, ambas entradas están a nivel BAJO (0), luego en este intervalo la salida estará a nivel BAJO (0). Durante el intervalo t_4, la salida está a nivel ALTO (1) ya que la entrada *A* está a nivel ALTO (1).

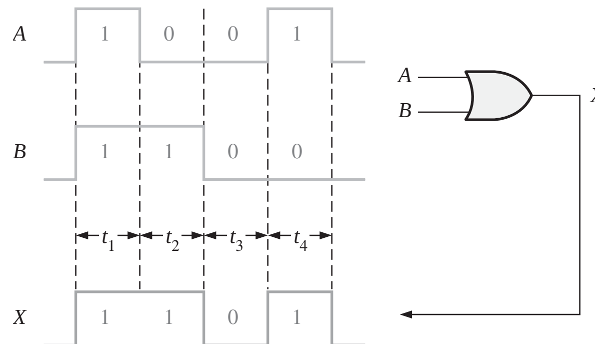


FIGURA 3.19 Ejemplo de funcionamiento de la puerta OR con trenes de impulsos junto con el cronograma que muestra la relación entre las entradas y la salida.

En este ejemplo, simplemente hemos aplicado la tabla de verdad de la puerta OR para cada uno de los intervalos de tiempo durante los cuales los niveles no cambiaban. Los ejemplos del 3.6 al 3.8 ilustran el funcionamiento de la puerta OR con diferentes señales de entrada.

EJEMPLO 3.6

Si se aplican las dos señales de entrada, *A* y *B*, de la Figura 3.20 a la puerta OR, ¿cuál es la señal de salida resultante?

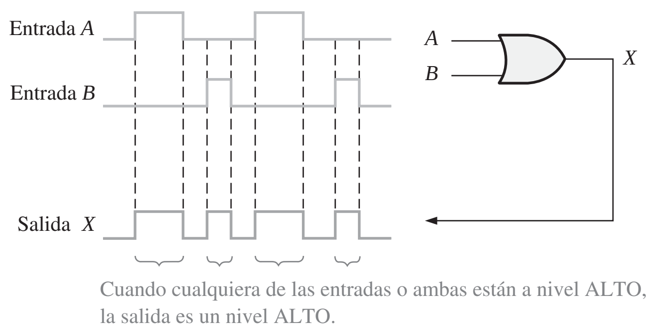


FIGURA 3.20

Solución La señal de salida *X* de una puerta OR de dos entradas será un nivel ALTO cuando una o ambas entradas estén a nivel ALTO, tal como muestra el diagra-

ma de tiempos. En este caso, ambas entradas nunca están al tiempo a nivel ALTO.

Problema relacionado Determinar la señal de salida y dibujar el diagrama de tiempos si la entrada *A* se cambia de modo que está a nivel ALTO desde el inicio del primer impulso hasta el final del segundo impulso.

EJEMPLO 3.7

Para las dos ondas de entrada, *A* y *B*, de la Figura 3.21, dibujar la onda de salida indicando su relación respecto a las entradas.

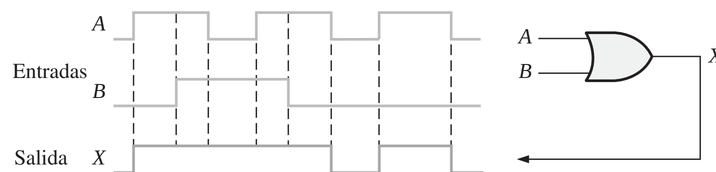


FIGURA 3.21

Solución Cuando una o ambas entradas están a nivel ALTO, la salida estará a nivel ALTO, como muestra la señal de salida *X* en el diagrama de tiempos.

Problema relacionado Determinar la señal de salida y dibujar el cronograma si el impulso central de la entrada *A* se sustituye por un nivel BAJO.

EJEMPLO 3.8

Para la puerta OR de 3 entradas de la Figura 3.22, determinar la señal de salida respecto de las entradas en función del tiempo.

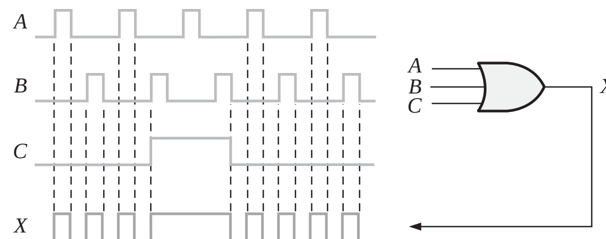


FIGURA 3.22

Solución La salida está a nivel ALTO cuando una o más entradas están a nivel ALTO, como muestra la señal de salida *X* en el diagrama de tiempos.

Problema relacionado Determinar la señal de salida y dibujar el diagrama de tiempos si la entrada *C* está siempre a nivel BAJO.

Expresiones lógicas de la puerta OR

▲ *Cuando las variables están separadas mediante el símbolo +, se aplica la operación OR.*

La función lógica OR de dos variables se representa matemáticamente mediante un signo + entre las dos variables, por ejemplo, $A + B$.

La suma en el álgebra de Boole implica variables cuyos valores son o el binario 1 o el binario 0. Las reglas básicas de la **suma booleana** son las siguientes:

$$0 + 0 = 0$$
$$0 + 1 = 1$$
$$1 + 0 = 1$$
$$1 + 1 = 1$$

La suma booleana es lo mismo que la función OR.

Observe que la suma booleana difiere de la suma binaria en el caso en que se suman dos 1s. En la suma booleana no existe acarreo.

El funcionamiento de una puerta OR de 2 entradas se puede expresar como sigue: si una variable de entrada es A y la otra variable de entrada es B, y la variable de salida es X, entonces la expresión booleana es

$$X = A + B$$

La Figura 3.23(a) muestra el símbolo lógico de la puerta indicando las variables de entradas y de salida.

Para ampliar la expresión OR a más de dos variables de entrada, se usa una nueva letra para cada variable adicional. Por ejemplo, la función de una puerta OR de 3 entradas se puede expresar como $X = A + B + C$, y la expresión para una puerta OR de 4 entradas sería $X = A + B + C + D$, y así sucesivamente. Las ilustraciones (b) y (c) de la Figura 3.23 muestran, respectivamente, las variables de entrada de las puertas OR de tres y cuatro entradas.

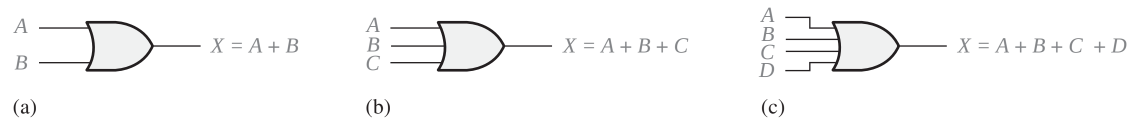
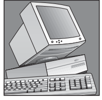


FIGURA 3.23 Expresiones booleanas de las puertas OR con dos, tres y cuatro entradas.

Como se muestra en la Tabla 3.6 para una puerta OR de dos entradas, el funcionamiento de la puerta OR se puede evaluar utilizando las expresiones booleanas para la salida X, sustituyendo todas las posibles combinaciones de 1 y 0 en las variables de entrada. Esta evaluación muestra que la salida X de una puerta OR es un 1 (nivel ALTO) cuando una o más de sus entradas son 1 (nivel ALTO). Un análisis similar se puede hacer para las puertas OR con cualquier número de variables de entrada.

A	B	$A + B = X$
0	0	0 + 0 = 0
0	1	0 + 1 = 1
1	0	1 + 0 = 1
1	1	1 + 1 = 1

TABLA 3.6



NOTAS INFORMÁTICAS

Otra operación de enmascaramiento que se usa en la programación de computadoras para hacer que determinados bits de un byte de datos sean igual a 1 (lo que se denomina activación) de forma selectiva, no afectando a otros bits, se lleva a cabo con la operación OR. Se utiliza una máscara que contenga un 1 en cualquier posición en la que un bit de datos desee ponerse a 1. Por ejemplo, si desea forzar que el bit más significativo de un byte de datos sea igual a 1, dejando los demás bits como están, puede aplicar la operación OR al byte de datos con la máscara 10000000.

Aplicación

En la Figura 3.24 se presenta parte de un sistema de alarma y detección de intrusos simplificado. Este sistema se podría utilizar en una habitación de una casa: una habitación con dos ventanas y una puerta. Los sensores son interruptores magnéticos que producen un nivel de salida ALTO cuando se abre la puerta (o las ventanas) y una salida a nivel BAJO cuando se cierra. Mientras que las ventanas y la puerta están aseguradas, los interruptores están cerrados y las tres entradas de la puerta OR están a nivel BAJO. Cuando se abre una de las ventanas o la puerta, en la entrada correspondiente de la puerta OR se genera un nivel ALTO y la salida de la puerta lógica se pone a nivel ALTO. Entonces se activa el circuito de alarma para advertir de la intrusión.

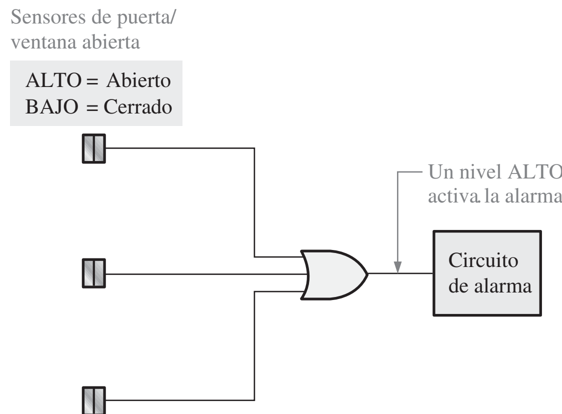


FIGURA 3.24 Un sistema de detección de intrusión simplificado que utiliza una puerta OR.

REVISIÓN DE LA SECCIÓN 3.3

1. ¿Cuándo se pone a nivel ALTO la salida de una puerta OR?
2. ¿Cuándo se pone a nivel BAJO la salida de una puerta OR?
3. Escribir la tabla de verdad para una puerta OR de tres entradas.

3.4 LA PUERTA NAND

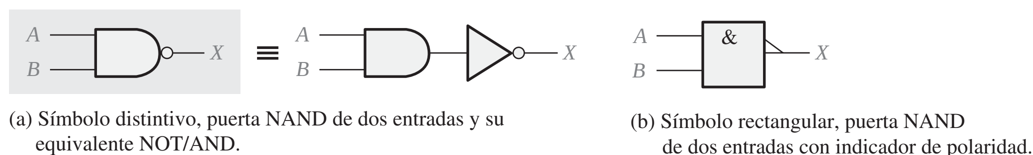
La puerta NAND es un elemento lógico popular, debido a que se puede utilizar como una puerta universal, es decir, las puertas NAND se pueden combinar para implementar las operaciones de las puertas AND, OR y del inversor. En el Capítulo 5, se examinará la propiedad universal de la puerta NAND.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Identificar una puerta NAND mediante su símbolo distintivo y su símbolo rectangular. ■ Describir la operación lógica de una puerta NAND. ■ Desarrollar la tabla de verdad de una puerta NAND con

cualquier número de entradas. ■ Generar el cronograma de una puerta NAND para cualquier forma de onda especificada en sus entradas. ■ Escribir la expresión lógica de una puerta NAND con cualquier número de entradas. ■ Describir la operación de la puerta NAND en términos de la puerta equivalente negativa–OR. ■ Desarrollar ejemplos de aplicaciones de la puerta NAND.

El término *NAND* es una contracción de NOT−AND, e implica una función AND con la salida complementada (negada). En la Figura 3.25(a) se muestra el símbolo lógico estándar para la puerta NAND de 2 entradas y su equivalente empleando los símbolos de la puerta AND seguida de un inversor, donde el símbolo ≡ significa "equivalente a". En la parte (b) se muestra el símbolo rectangular.



(a) Símbolo distintivo, puerta NAND de dos entradas y su equivalente NOT/AND.

(b) Símbolo rectangular, puerta NAND de dos entradas con indicador de polaridad.

FIGURA 3.25 Símbolos lógicos estándar de la puerta NAND (ANSI–/IEEE 91–1984).

Funcionamiento de la puerta NAND

La ***puerta NAND*** genera una salida a nivel BAJO sólo cuando todas las entradas están a nivel ALTO. Cuando cualquiera de las entradas está a nivel BAJO, la salida se pondrá a nivel ALTO. Para el caso concreto de la puerta NAND de dos entradas, como la mostrada en la Figura 3.25, con la designación *A* y *B* para las entradas y *X* para la salida, el modo de operación se puede establecer como sigue:

En una puerta NAND de dos entradas, la salida *X* es un nivel BAJO si las entradas *A* y *B* están a nivel ALTO; *X* es un nivel ALTO si *A* o *B* están a nivel BAJO o si ambas, *A* y *B*, están a nivel BAJO.

Observe que esta operación, en términos de nivel de salida, es la opuesta a la operación lógica AND. En una puerta NAND, el nivel BAJO (0) es el nivel activo o verdadero de salida, como indica el círculo de la salida. La Figura 3.26 ilustra la operación lógica de una puerta NAND de dos entradas, para las cuatro posibles combinaciones, y la Tabla 3.7 es la tabla de verdad, que resume la operación lógica de la puerta NAND de dos entradas.



FIGURA 3.26 Funcionamiento de la puerta NAND de 2 entradas.

Funcionamiento con trenes de impulsos

Veamos ahora el funcionamiento de la puerta NAND con un tren de impulsos. Recuerde de su tabla de verdad que la salida se pone a nivel BAJO sólo cuando todas las entradas están a nivel ALTO.

Operación equivalente negativa-OR de la puerta NAND. Es inherente al funcionamiento de la puerta NAND el hecho de que una o más entradas a nivel BAJO dan lugar a una salida a nivel ALTO. La Tabla 3.7 indica que

Entradas		Salida
A	*B*	*X*
0	0	1
0	1	1
1	0	1
1	1	0
1 = ALTO, 0 = BAJO		

TABLA 3.7 Tabla de verdad de la puerta NAND de 2 entradas.

EJEMPLO 3.9

Si a las entradas de una puerta NAND se aplican las formas de onda *A* y *B* de la Figura 3.27, determinar la forma de onda resultante de salida.

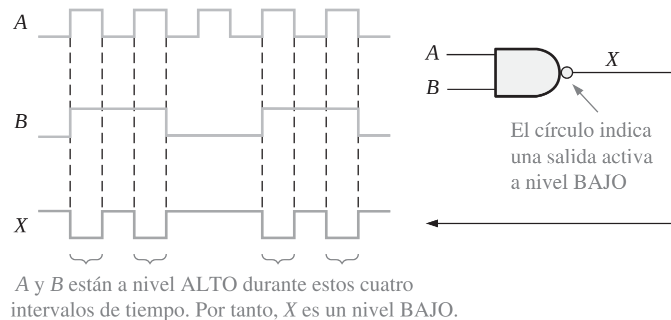


FIGURA 3.27

Solución La señal de salida *X* está a nivel BAJO sólo durante cuatro intervalos de tiempo, cuando ambas entradas, *A* y *B*, están a nivel ALTO, como muestra el diagrama de tiempos.

Problema relacionado Determinar la onda de salida y dibujar el diagrama de tiempos si se invierte la entrada *B*.

la salida *X* está a nivel ALTO (1) cuando cualquiera de las entradas, *A* y *B*, están a nivel BAJO (0). Desde este punto de vista, la puerta NAND se puede usar para realizar la operación OR que requiere una o más entradas a nivel BAJO, para generar una salida a nivel ALTO. Este modo de operación se denomina **negativa–OR**. En este contexto, el término *negativa* significa que las entradas se definen para que su estado activo o verdadero sea un nivel BAJO.

En una puerta NAND de dos entradas que funciona como una puerta negativa–OR, la salida *X* es un nivel ALTO si cualquiera de las entradas, *A* o *B*, es un nivel BAJO, o si ambas entradas, *A* y *B*, están a nivel BAJO.

Cuando la puerta NAND se emplea con uno o más niveles bajos en sus entradas en lugar de con niveles altos, se trata como una puerta negativa–OR y se representa con el símbolo lógico estándar de la Figura 3.29.

EJEMPLO 3.10

Obtener la forma de onda de salida para la puerta NAND de tres entradas de la Figura 3.28, donde además se presenta el diagrama de tiempos de las entradas.

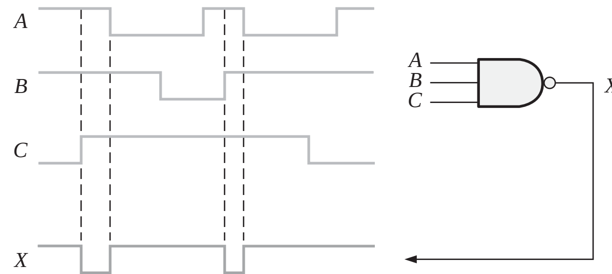


FIGURA 3.28

Solución La señal de salida X está a nivel BAJO sólo cuando las tres entradas están a nivel ALTO, tal como se muestra en el diagrama de tiempos.

Problema relacionado Determinar la forma de onda de salida y dibujar el diagrama de tiempos si se invierte la señal de entrada A.

Aunque los dos símbolos de esta figura representan la misma puerta física, sirven para definir el papel o el modo de operación en una aplicación particular, como se ilustra en los Ejemplos 3.11 hasta 3.13.

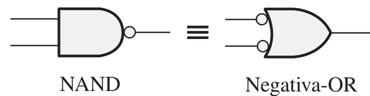


FIGURA 3.29 Símbolos estándar para representar las dos operaciones equivalentes de la puerta NAND.

EJEMPLO 3.11

Una planta de fabricación utiliza dos tanques para almacenar un determinado líquido químico que se requiere en un proceso de fabricación. Cada tanque dispone de un sensor que detecta cuándo el nivel del líquido cae al 25% del total. Los sensores generan una tensión de 5 V cuando los tanques están llenos por encima del 25%. Cuando el volumen de líquido en el tanque cae por debajo del 25%, el sensor genera un nivel de 0 V.

En el panel indicador se requiere un diodo emisor de luz (LED, *Light–Emitting Diode*) verde que indique que el nivel de ambos tanques está por encima del 25%. Como se indica, se puede utilizar una puerta NAND para implementar esta función.

Solución La Figura 3.30 muestra una puerta NAND de dos entradas conectadas a los sensores de nivel del tanque y la salida conectada al panel indicador. La operación puede definirse como sigue: si el nivel del tanque A y el nivel del tanque B están por encima de un cuarto del total, el LED se enciende.

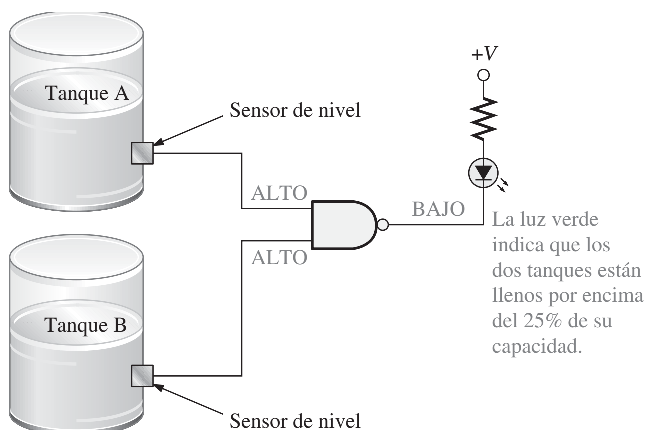


FIGURA 3.30

Mientras que la salida de *ambos* sensores esté a nivel ALTO (5 V), quiere decir que el nivel de los tanques está por encima del 25% del volumen total, y la salida de la puerta NAND está a nivel BAJO (0 V). El circuito del LED verde se activa para una tensión a nivel BAJO.

Problema relacionado ¿Cómo se puede modificar el circuito de la Figura 3.30 para controlar el nivel de tres tanques en lugar de dos?

EJEMPLO 3.12

El supervisor del proceso de fabricación descrito en el Ejemplo 3.11 ha decidido que sería preferible disponer de un LED rojo encendido cuando al menos el nivel de líquido de uno de los tanques estuviera por debajo del 25%, y que el LED verde se encendiera cuando el nivel en ambos tanques estuviera por encima de dicho límite. Veamos cómo se puede implementar este requisito.

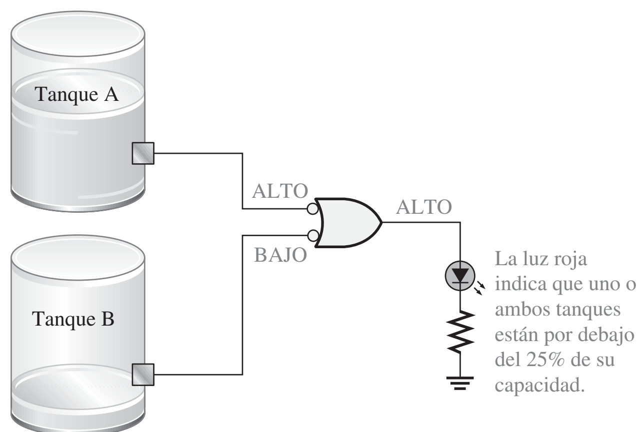


FIGURA 3.31

Solución La Figura 3.31 muestra que la puerta NAND está funcionando como una puerta negativa–OR, para detectar la ocurrencia de que al menos una de las entradas esté a nivel BAJO. Un sensor genera un nivel BAJO si el volumen en su tanque es del 25% o menor. Cuando esto ocurre, la salida de la puerta pasa a nivel ALTO, por lo que el LED rojo del panel indicador se enciende para un nivel de tensión ALTO. La operación puede definirse del siguiente modo: si el tanque A o el tanque B, *o* ambos están por debajo del 25% del total, entonces el LED se enciende.

Observe que en este ejemplo y en el Ejemplo 3.11, se usa la misma puerta NAND de dos entradas, pero en el esquema se ha empleado un símbolo de puerta diferente para ilustrar la diferencia funcional de las puertas NAND y negativa–OR.

Problema relacionado ¿Cómo se puede modificar el circuito de la Figura 3.31 para controlar cuatro tanques en lugar de dos?

EJEMPLO 3.13

Para la puerta NAND de cuatro entradas de la Figura 3.32, que opera como una puerta negativa–OR, determinar la salida con respecto a sus entradas.

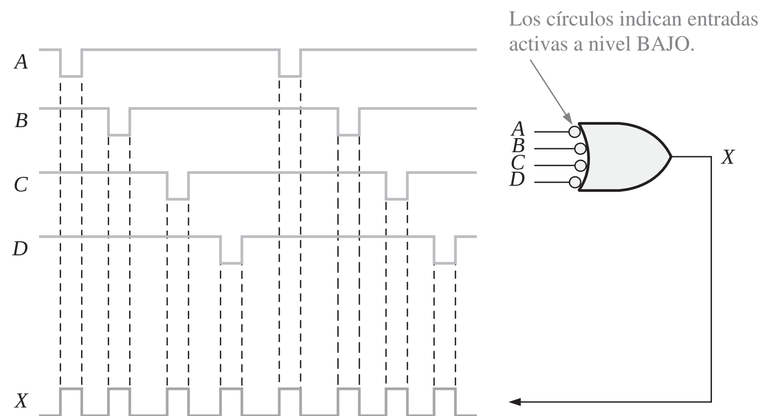


FIGURA 3.32

Solución La forma de onda de salida X estará a nivel ALTO siempre que una entrada esté a nivel BAJO, como se indica en el diagrama de tiempos.

Problema relacionado Determinar la forma de onda de salida, si se invierte la señal de entrada A antes de aplicarla a la puerta.

Expresiones lógicas para la puerta NAND

La expresión booleana para la puerta NAND de dos entradas es:

$$X = \overline{AB}$$

Esta expresión quiere decir que las dos variables de entrada, A y B, se multiplican (AND) primero y luego se complementan, tal y como indica la barra sobre la expresión lógica correspondiente a AND. Ésta es una descripción lógica en forma de ecuación de la operación de una puerta NAND con dos entradas. Si evalúa esta expresión para todos los posibles valores de las dos variables de entrada, el resultado que se obtiene es el indicado en la Tabla 3.8.

▲ *Una barra encima de una o varias variables indica una operación de inversión.*

A	B	$\overline{AB} = X$
0	0	$\overline{0 \cdot 0} = \overline{0} = 1$
0	1	$\overline{0 \cdot 1} = \overline{0} = 1$
1	0	$\overline{1 \cdot 0} = \overline{0} = 1$
1	1	$\overline{1 \cdot 1} = \overline{1} = 0$

TABLA 3.8

Una vez determinada la expresión lógica para una función lógica dada, esta función se puede evaluar para todos los valores posibles de las variables. La evaluación facilita de forma exacta la salida del circuito lógico para cada una de las condiciones de entrada y, por tanto, le proporciona una descripción completa de la operación lógica del circuito. La expresión para NAND se puede extender a más de dos variables de entrada, incluyendo letras adicionales para representar las otras variables.

REVISIÓN DE LA SECCIÓN 3.4

1. ¿Cuándo está a nivel BAJO la salida de una puerta NAND?
2. ¿Cuándo está a nivel ALTO la salida de una puerta NAND?
3. Describir las diferencias funcionales entre una puerta NAND y una puerta negativa–OR. ¿Tienen ambas la misma tabla de verdad?
4. Escribir la expresión de salida para una puerta NAND con tres entradas, A, B y C.

3.5 LA PUERTA NOR

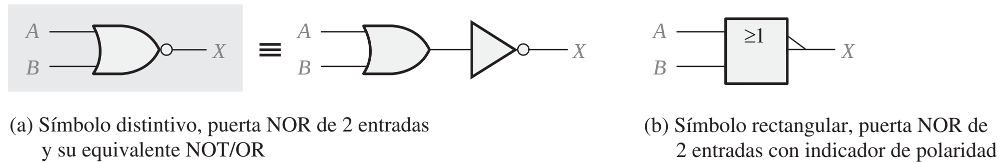
La puerta NOR, al igual que la puerta NAND, es un útil elemento lógico porque también se puede emplear como una puerta universal; es decir, las puertas NOR se pueden usar en combinación para implementar las operaciones AND, OR y del inversor. En el Capítulo 5 se examinará la propiedad universal de la puerta NOR.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Identificar una puerta NOR mediante su símbolo distintivo y su símbolo rectangular. ■ Describir el funcionamiento de una puerta NOR. ■ Desarrollar la tabla de verdad de una puerta NOR con cualquier número de entradas. ■ Generar el diagrama de tiempos de una puerta NOR para cualquier forma de onda especificada en sus entradas. ■ Escribir la expresión lógica de una puerta NOR con cualquier número de entradas. ■ Describir el funcionamiento de la puerta NOR en términos de su equivalente negativa–AND. ■ Desarrollar ejemplos de aplicaciones de la puerta NOR.

▲ *NOR es igual que OR excepto porque la salida está invertida.*

El término *NOR* es una contracción de NOT−OR e implica una función OR con la salida invertida (complementada). En la Figura 3.33(a) se muestra el símbolo lógico estándar para la puerta NOR de 2 entradas y su equivalente empleando los símbolos de la puerta OR seguida de un inversor. En la parte (b) se muestra el símbolo rectangular.



(a) Símbolo distintivo, puerta NOR de 2 entradas y su equivalente NOT/OR

(b) Símbolo rectangular, puerta NOR de 2 entradas con indicador de polaridad

FIGURA 3.33 Símbolo lógico estándar para la puerta NOR (ANSI/IEEE Std. 91–1984)

Funcionamiento de la puerta NOR

La ***puerta NOR*** genera una salida a nivel BAJO cuando *cualquiera* de sus entradas está a nivel ALTO. Sólo cuando todas sus entradas estén a nivel BAJO, la salida se pondrá a nivel ALTO. Para el caso concreto de la puerta NOR de dos entradas, que se muestra en la Figura 3.33, con la designación *A* y *B* para las entradas y *X* para la salida, el modo de operación se puede establecer como sigue:

En una puerta NOR de dos entradas: la salida *X* es un nivel BAJO si cualquiera de sus entradas *A* o *B* está a nivel ALTO, o si ambas entradas *A* y *B* están a nivel ALTO; *X* es un nivel ALTO si *A* y *B* están a nivel BAJO.

Esta operación genera un nivel de salida opuesto al que genera la puerta OR. En una puerta NOR, el nivel BAJO es el nivel activo o verdadero de salida, como indica el círculo de la salida. La Figura 3.34 ilustra el funcionamiento de una puerta NOR de dos entradas, para las cuatro posibles combinaciones de entrada, y la Tabla 3.9 es la tabla de verdad para la puerta NOR de dos entradas.



FIGURA 3.34 Funcionamiento de la puerta NOR de 2 entradas.

Entradas *A*	*B*	Salida *X*
0	0	1
0	1	0
1	0	0
1	1	0
1 = ALTO, 0 = BAJO		

TABLA 3.9 Tabla de verdad de una puerta NOR de 2 entradas.

Funcionamiento con trenes de impulsos

Los dos ejemplos siguientes ilustran la operación lógica de la puerta NOR con entradas que son trenes de impulsos. De nuevo, como en los demás tipos de puertas, simplemente deberemos aplicar la tabla de verdad para determinar la forma de onda de salida de acuerdo con el diagrama de tiempos de las entradas.

EJEMPLO 3.14

Si se aplican a la puerta NOR las dos señales mostradas en la Figura 3.35, ¿cómo es la señal de salida que se obtiene?

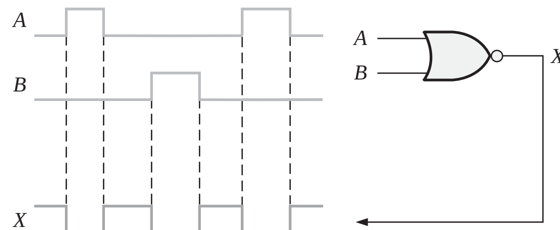


FIGURA 3.35

Solución

Cuando cualquier entrada de la puerta NOR está a nivel ALTO, la salida es un nivel BAJO, como se ve en la forma de onda de salida X del diagrama de tiempos.

Problema relacionado Invertir la entrada B y determinar la forma de onda de salida para esas entradas.

EJEMPLO 3.15

Determinar la señal de salida para la puerta NOR de tres entradas de la Figura 3.36, teniendo en cuenta el diagrama de tiempos correspondiente a las entradas.

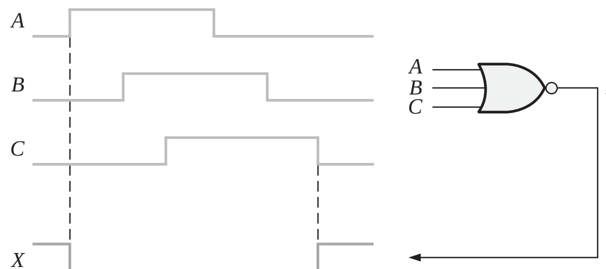


FIGURA 3.36

Solución La salida X se pone a nivel BAJO cuando cualquier entrada está a nivel ALTO, como muestra la señal de salida X en el diagrama de tiempos.

Problema relacionado Con las entradas B y C invertidas, determinar la salida y dibujar el diagrama de tiempos.

Operación equivalente negativa-AND de la puerta NOR. La puerta NOR, al igual que la puerta NAND, posee otro aspecto de su funcionamiento que es inherente a su función lógica. La Tabla 3.9 muestra que se genera un nivel ALTO en la salida sólo si todas las entradas están a nivel BAJO. Desde este punto de vista, la puerta NOR se puede utilizar como una operación AND cuyas entradas están a nivel BAJO y generan una salida a nivel ALTO. Este modo de operación se denomina **negativa–AND**. En este contexto, el término *negativa* significa que las entradas están definidas para que su estado activo o verdadero sea un nivel BAJO.

En una puerta NOR de dos entradas que funciona como una puerta negativa–AND, la salida *X* es un nivel ALTO cuando ambas entradas, *A* y *B*, están a nivel BAJO.

Cuando se quiere que la puerta NOR presente todas sus entradas a nivel BAJO en lugar de presentar uno o más niveles altos, se aplica la puerta negativa–AND y se representa mediante el símbolo estándar de la Figura 3.37. Es importante recordar que los dos símbolos de la Figura 3.37 representan la misma puerta física, y sólo sirven para distinguir entre los dos modos de operación lógicos. Los tres ejemplos siguientes ilustran esto.

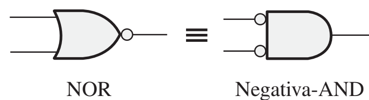


FIGURA 3.37 Símbolos estándar que representan dos operaciones equivalentes de la puerta NOR.

EJEMPLO 3.16

Se necesita un dispositivo para indicar cuándo se producen simultáneamente dos niveles de entrada bajos que dan lugar a un nivel de salida ALTO. Especificar el dispositivo.

Solución Para generar una salida a nivel ALTO cuando ambas entradas están a nivel BAJO se requiere una puerta negativa–AND, como se muestra en la Figura 3.38.



FIGURA 3.38

Problema relacionado Se necesita un dispositivo para indicar una salida a nivel BAJO, cuando uno o dos niveles altos se aplican a sus entradas. Especificar el dispositivo.

EJEMPLO 3.17

Como parte del sistema de monitorización funcional de un avión, se requiere un circuito para indicar el estado del tren de aterrizaje antes de tomar tierra. Se enciende un LED verde si los tres mecanismos de aterrizaje están correctamente extendidos cuando el interruptor para “bajar el tren de aterrizaje” se ha activado. Un LED rojo se enciende si cualquiera de los mecanismos falla al extenderse antes de aterrizar. Cuando uno de los mecanismos se extiende, el sensor correspondiente genera una tensión a nivel BAJO. Cuando uno de los mecanismos del tren de aterrizaje se retrae, su sensor genera una tensión a nivel ALTO. Implementar un circuito que cumpla estos requisitos.

Solución La alimentación se aplica al circuito sólo cuando el interruptor para "bajar el tren de aterrizaje" se activa. Se debe utilizar una puerta NOR para cada uno de los dos requisitos, tal y como se muestra en la Figura 3.39. Una puerta NOR funciona como una puerta negativa–AND para detectar un nivel BAJO procedente de cada uno de los sensores de los mecanismos de aterrizaje. Cuando las tres entradas de la puerta están a nivel BAJO, los tres mecanismos (ruedas) están correctamente extendidos y dan lugar a una salida a nivel ALTO en la puerta negativa–AND, lo que hace que se encienda el LED verde. La otra puerta NOR funciona como tal para detectar si una o más de las ruedas de aterrizaje permanecen retraídas cuando se activa el interruptor que baja el tren de aterrizaje. Cuando una o más de las ruedas de aterrizaje permanecen retraídas; la salida a nivel ALTO procedente del sensor se detecta mediante la puerta NOR, que da lugar a una salida a nivel BAJO que enciende el LED rojo de aviso.

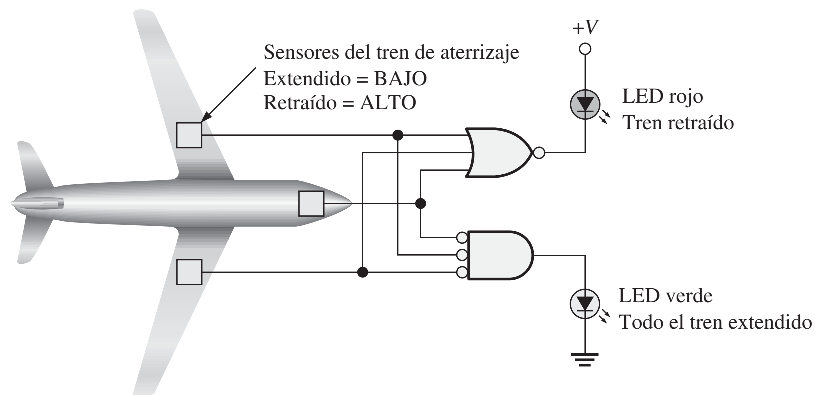


FIGURA 3.39

Problema relacionado ¿Qué tipo de puerta se debería usar para detectar si las tres ruedas del tren de aterrizaje están escondidas después de despegar? Suponer que se precisa un nivel de salida BAJO para activar un LED.

CONSEJOS PRÁCTICOS

Cuando se excita una carga tal como un LED con una puerta lógica, debería consultarse la hoja de características del fabricante para conocer los parámetros máximos de excitación (corriente de salida). Un circuito integrado de puerta lógica normal puede no ser capaz de manipular la corriente necesaria para determinadas cargas, como por ejemplo algunos LED. Hay disponibles puertas lógicas con salida tipo buffer, tal como una salida en colector abierto o en drenador abierto en muchos tipos de configuraciones de circuitos integrados. La capacidad de corriente de salida de las puertas lógicas en CI típicas están limitadas en el rango de los µA o de los mA. Por ejemplo, las puertas estándar TTL pueden manipular corrientes de salida de hasta 16 mA. La mayor parte de los LED precisan corrientes en el rango comprendido entre los 10 mA y 50 mA.

EJEMPLO 3.18

Para una puerta NOR de cuatro entradas que opera como una puerta negativa–AND como la de la Figura 3.40, determinar la salida en función de las entradas.

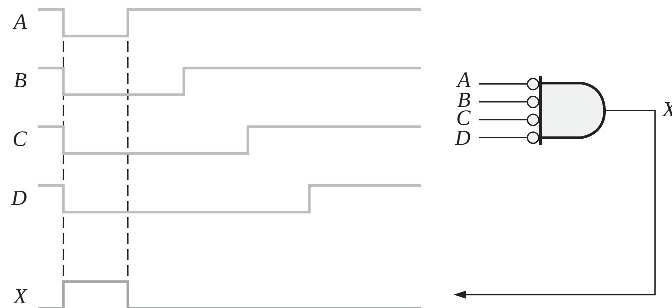


FIGURA 3.40

Solución En cualquier instante en el que todas las entradas estén a nivel BAJO, la salida estará a nivel ALTO, siendo la forma de onda de salida *X* la indicada en el cronograma.

Problema relacionado Determinar la salida para la entrada *D* invertida y dibujar el diagrama de tiempos.

Expresiones lógicas para la puerta NOR

La expresión booleana para la salida de una puerta NOR de dos entradas se expresa así:

$$X = \overline{A + B}$$

Esta ecuación indica que las dos variables de entrada primero se suman (operación OR) y luego se complementan, tal y como indica la barra sobre la expresión lógica OR. Evaluando esta expresión, se obtienen los resultados mostrados en la Tabla 3.10. La expresión NOR puede extenderse a más de dos variables de entrada, incluyendo letras adicionales para representar otras variables.

A	***B***	$\overline{A+B} = X$
0	0	$\overline{0+0} = \overline{0} = 1$
0	1	$\overline{0+1} = \overline{1} = 0$
1	0	$\overline{1+0} = \overline{1} = 0$
1	1	$\overline{1+1} = \overline{1} = 0$

TABLA 3.10

REVISIÓN DE LA SECCIÓN 3.5

1. ¿Cuándo está a nivel ALTO la salida de una puerta NOR?
2. ¿Cuándo está a nivel BAJO la salida de una puerta NOR?

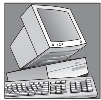
3. Describir la diferencia funcional entre una puerta NOR y una puerta negativa–AND. ¿Tienen ambas la misma tabla de verdad?
4. Escribir las expresiones de salida para una puerta NOR de tres entradas, siendo las variables de entrada *A*, *B* y *C*.

3.6 PUERTAS OR–EXCLUSIVA Y NOR–EXCLUSIVA

Las puertas OR–exclusiva y NOR–exclusiva se forman mediante la combinación de otras puertas que ya hemos tratado, como se verá en el Capítulo 5. Sin embargo, debido a su importancia fundamental en muchas aplicaciones, estas puertas se tratan como elementos lógicos básicos con su propio símbolo exclusivo.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Identificar las puertas OR–exclusiva y NOR–exclusiva mediante su símbolo distintivo y su símbolo rectangular. ■ Describir la operación lógica de las puertas OR–exclusiva y NOR–exclusiva. ■ Desarrollar las tablas de verdad de las puertas OR–exclusiva y NOR–exclusiva. ■ Generar el diagrama de tiempos de las puertas OR–exclusiva y NOR–exclusiva para cualquier forma de onda especificada en sus entradas. ■ Desarrollar ejemplos de aplicaciones de las puertas NOR–exclusiva y OR–exclusiva.



NOTAS INFORMÁTICAS

Varias puertas OR–exclusivas conectadas para formar un circuito sumador permiten que una computadora realice las operaciones de suma, sustracción, multiplicación y división en su unidad aritmético lógica (ALU). Una puerta OR–exclusiva se implementa con los circuitos lógicos básicos de las puertas AND, OR y NOT.

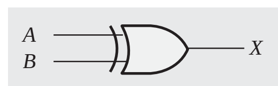
La puerta OR–exclusiva

▲ *En una puerta OR–exclusiva, entradas opuestas proporcionan una salida a nivel ALTO.*

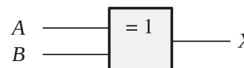
En la Figura 3.41 se muestran los símbolos estándar para la puerta OR–exclusiva (XOR). La puerta XOR tiene sólo dos entradas.

La salida de una ***puerta OR–exclusiva*** se pone a nivel ALTO *sólo* cuando las dos entradas están a niveles lógicos opuestos. Esta operación se puede expresar, en función de dos entradas *A* y *B* y una salida *X*, del siguiente modo:

En una puerta OR–exclusiva, la salida *X* es un nivel ALTO si la entrada *A* está a nivel BAJO y la entrada *B* está a nivel ALTO; o si la entrada *A* está a nivel ALTO y la entrada *B* está a nivel BAJO; *X* es un nivel BAJO si tanto *A* como *B* están a nivel ALTO o BAJO.



(a) Símbolo distintivo



(b) Símbolo rectangular con la puerta XOR

FIGURA 3.41 Símbolos lógicos estándar de la puerta OR–exclusiva.

En la Figura 3.42 se ilustran las cuatro posibles combinaciones de entrada y las salidas resultantes para la puerta XOR. El nivel ALTO es el nivel activo o verdadero de salida y sólo se produce cuando las entradas

están a niveles opuestos. La operación lógica de la puerta XOR se resume en la tabla de verdad mostrada en la Tabla 3.11.

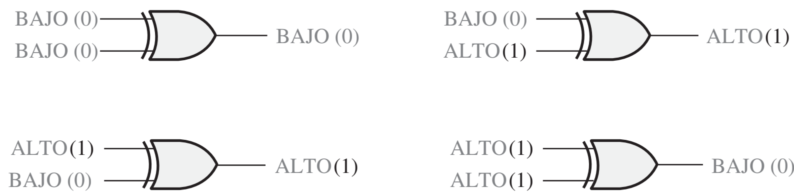


FIGURA 3.42 Todos los niveles lógicos posibles para una puerta OR–exclusiva.

Entradas *A*	*B*	Salida *X*
0	0	0
0	1	1
1	0	1
1	1	0

TABLA 3.11 Tabla de verdad de la puerta OR–exclusiva

EJEMPLO 3.19

Un cierto sistema está formado por dos circuitos idénticos que funcionan en paralelo. Mientras que ambos funcionan correctamente, las salidas de los dos circuitos son iguales. Si uno de los circuitos falla, las salidas serán niveles opuestos en ese instante. Establecer un método para detectar que se ha producido un fallo en uno de los circuitos.

Solución Las salidas de los circuitos se conectan a las entradas de una puerta XOR, tal y como muestra la Figura 3.43. Un fallo en cualquiera de los circuitos hace que las entradas de la puerta XOR tengan niveles opuestos. Esta condición da lugar a nivel ALTO en la salida de la puerta XOR, que indica que uno de los circuitos ha fallado.

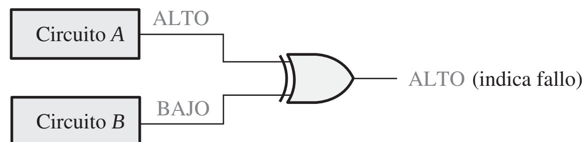


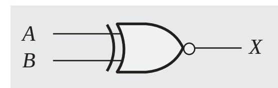
FIGURA 3.43

Problema relacionado ¿Detectará siempre la puerta OR–exclusiva fallos simultáneos en los dos circuitos de la Figura 3.43? Si la respuesta es no, ¿bajo qué condiciones los detectará?

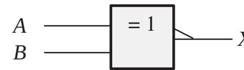
La puerta NOR-exclusiva

En la Figura 3.44 se presentan los símbolos estándar de la **puerta NOR–exclusiva** (XNOR). Al igual que la puerta XOR, la puerta XNOR sólo tiene dos entradas. El círculo en la salida del símbolo de la puerta XNOR indica que su salida es la opuesta a la de la puerta XOR. Cuando dos niveles lógicos de entrada son opuestos, la salida de la puerta NOR–exclusiva es un nivel BAJO. La operación se puede expresar del siguiente modo (*A* y *B* son las entradas, y *X* es la salida).

En una puerta NOR–exclusiva, la salida *X* es un nivel BAJO si la entrada *A* está a nivel BAJO y la entrada *B* está a nivel ALTO, o si *A* está a nivel ALTO y *B* está a nivel BAJO; *X* es un nivel ALTO si *A* y *B* están ambas a nivel ALTO o BAJO.



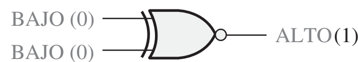
(a) Símbolo distintivo



(b) Símbolo rectangular

FIGURA 3.44 Símbolos lógicos estándar para la puerta NOR–exclusiva.

En la Figura 3.45 se muestran las cuatro posibles combinaciones de entrada y las salidas resultantes para la puerta XNOR. La operación lógica se resume en la Tabla 3.12. Observe que la salida es un nivel ALTO cuando las dos entradas están al mismo nivel.



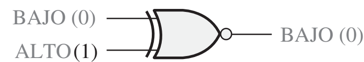






FIGURA 3.45 Todos los niveles lógicos posibles para una puerta NOR–exclusiva.

Entradas *A*	*B*	**Salida** *X*
0	0	1
0	1	0
1	0	0
1	1	1

Tabla 3.12 Tabla de verdad de la puerta NOR–exclusiva.

Funcionamiento con trenes de impulsos

Al igual que hemos hecho con las puertas anteriores, vamos a examinar el funcionamiento de las puertas XOR y XNOR para trenes de impulsos en sus entradas. Como antes, aplicamos la tabla de verdad para cada intervalo de tiempo de los trenes de impulsos, como se muestra en la Figura 3.46 para una puerta XOR. Puede comprobar que las señales de entrada *A* y *B* tienen niveles opuestos en los intervalos t_2 y t_4. Por tanto, la sali-

da X se pone a nivel ALTO durante estos dos intervalos. En los intervalos t_1 y t_3, ambas entradas están al mismo nivel, BAJO o ALTO, y la salida se pone a nivel BAJO para estos impulsos, como muestra el diagrama de tiempos.

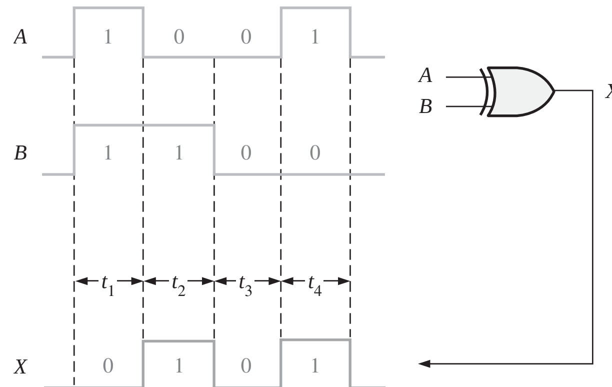


FIGURA 3.46 Ejemplo de funcionamiento de la puerta OR–exclusiva con trenes de impulsos.

EJEMPLO 3.20

Determinar las formas de onda de salida para la puerta XOR y la puerta XNOR, dadas las formas de onda de entrada A y B de la Figura 3.47.

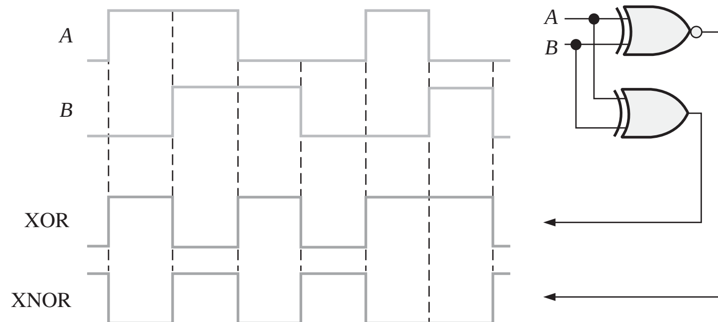


FIGURA 3.47

Solución Las formas de onda de salida son las que se indican en la Figura 3.47. Observe que la salida XOR está a nivel ALTO sólo cuando las entradas están a niveles opuestos. La salida XNOR está a nivel ALTO sólo cuando ambas entradas son iguales.

Problema relacionado Determinar la forma de onda de salida si las dos entradas, A y B, se invierten.

Aplicación

La puerta OR–exclusiva se puede utilizar como sumador de dos bits. Recuerde del Capítulo 2 que las reglas básicas de la suma binaria son: 0 + 0 = 0, 0 + 1 = 1, 1 + 0 = 1 y 1 +1 = 10. Un examen de la tabla de verdad

de la puerta XOR le demostrará que su salida es la suma binaria de los dos bits de entrada. En el caso en que ambas entradas sean 1, la salida de la suma es 0 y se pierde el acarreo de 1. En el Capítulo 6, verá cómo se combinan puertas XOR para implementar circuitos sumadores completos. La Figura 3.48 presenta la puerta XOR utilizada como sumador básico.

Bits de entrada		*Salida (suma)*
A	*B*	Σ
0	0	0
0	1	1
1	0	1
1	1	0 (sin acarreo 1)

FIGURA 3.48 Puerta XOR utilizada como sumador de dos bits.

REVISIÓN DE LA SECCIÓN 3.6

1. ¿Cuándo está a nivel ALTO la salida de una puerta XOR?
2. ¿Cuándo está a nivel ALTO la salida de una puerta XNOR?
3. ¿Cómo se puede utilizar una puerta XOR para detectar que dos bits son diferentes?

3.7 LÓGICA PROGRAMABLE

En el Capítulo 1 se ha presentado la lógica programable. En esta sección, se aborda el concepto básico de matriz AND programable, que constituye el pilar de la mayor parte de la lógica programable y se cubren las principales tecnologías de proceso. Un dispositivo lógico programable (PLD, *Programmable Logic Device*) es aquel que inicialmente no tiene una función lógica fija pero que puede programarse para implementar cualquier diseño lógico. Como ya hemos visto, los dos tipos de PLD son el SPLD y el CPLD. Además del PLD, la otra categoría importante de dispositivo lógico programable es la FPGA. Con el fin de simplificar, haremos referencia a todos estos dispositivos como dispositivos PLD. También se abordan algunos conceptos importantes sobre programación.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Describir el concepto de matriz AND programable. ■ Explicar distintas tecnologías de procesos. ■ Describir la introducción de datos en forma de texto y mediante gráficos como los dos métodos de diseño lógico programable. ■ Describir métodos para descargar un diseño en un dispositivo lógico programable. ■ Explicar la programación dentro del sistema.

Concepto básico de la matriz AND

La mayor parte de los tipos de dispositivo PLD utilizan alguna forma de ***matriz AND***. Esta matriz está formada, básicamente, por puertas AND y una matriz de interconexiones con conexiones programables en cada

punto de intersección, como se muestra en la Figura 3.49(a). El propósito de las conexiones programables es establecer o interrumpir una conexión entre una línea de fila y una línea de columna de la matriz de interconexión. Para cada entrada de una puerta AND, sólo se deja intacta una conexión programable con el fin de conectar la variable deseada a la entrada de la puerta. La Figura 3.49(b) muestra una matriz después de haber sido programada.

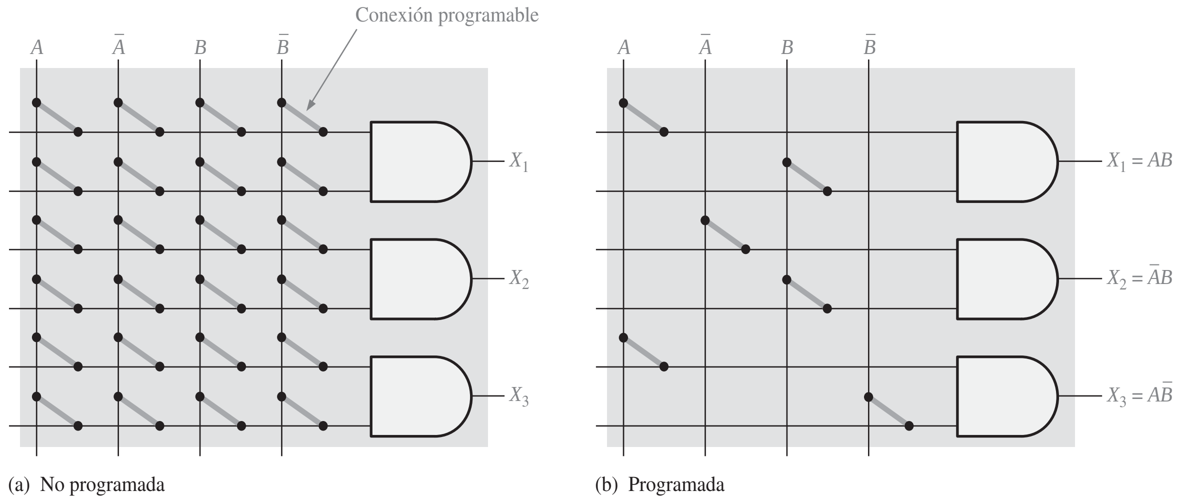


FIGURA 3.49 Concepto básico de una matriz AND programable.

EJEMPLO 3.21

Indicar cómo debe programarse la matriz AND de la Figura 3.49(a) para obtener las siguientes salidas: $X_1 = A\bar{B}$, $X_2 = \bar{A}B$ y $X_3 = \bar{A}\bar{B}$

Solución *Véase* la Figura 3.50.

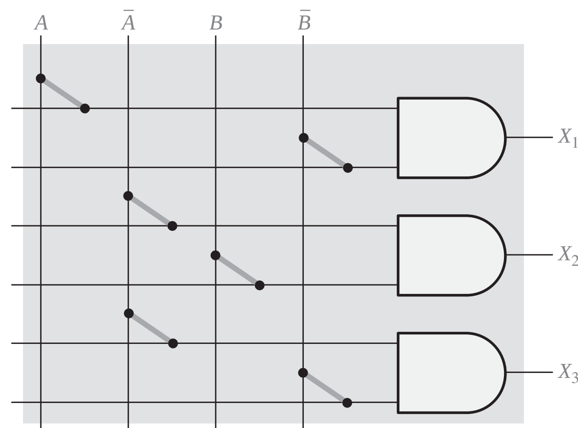


FIGURA 3.50

Problema relacionado	¿Cuántas filas, columnas y entradas de puerta AND son necesarias para tres variables de entrada en una matriz de puertas AND de 3 entradas?

Tecnologías de proceso basadas en conexiones programables

En los PLD se emplean varias tecnologías de proceso diferentes basadas en conexiones programables.

Tecnología basada en fusible. Ésta fue la tecnología original basada en conexiones programables, y todavía se emplea en algunos SPLD. El ***fusible*** es una conexión de metal que conecta una fila y una columna en la matriz de interconexión. Antes de realizar la programación, en cada intersección existe una conexión mediante fusible. Para programar un dispositivo, los fusibles seleccionados se abren haciendo pasar a su través una corriente que permita "fundir" el fusible y romper la conexión. Los fusibles que permanecen intactos proporcionan una conexión entre las filas y las columnas. En la Figura 3.51 se muestra una conexión mediante fusible. Los dispositivos lógicos programables que usan esta tecnología sólo pueden programarse una vez (**OTP,** *One–Time Programmable*).

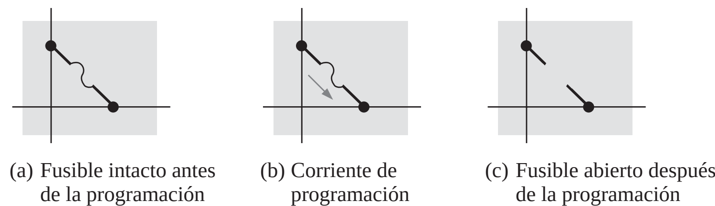


FIGURA 3.51 Conexión programable mediante fusible.

Tecnología basada en antifusible. Una conexión programable mediante **antifusible** es lo opuesto a una conexión mediante fusible. En lugar de interrumpir la conexión, durante la programación se establece una conexión. Inicialmente, un antifusible es un circuito abierto mientras que un fusible se comporta como un cortocircuito. Antes de llevar a cabo la programación, no existe ninguna conexión entre las filas y las columnas de la matriz de interconexión. Un antifusible está formado, básicamente, por dos conductores separados por un aislante. Para programar un dispositivo utilizando esta técnica, una herramienta de programación aplica una tensión adecuada a los antifusibles seleccionados para romper el aislamiento entre los dos conductores, de modo que el aislante pase a ser una conexión de baja resistencia. En la Figura 3.52 se muestra una conexión mediante antifusible. Los dispositivos antifusible también son dispositivos que sólo pueden programarse una vez (OTP).

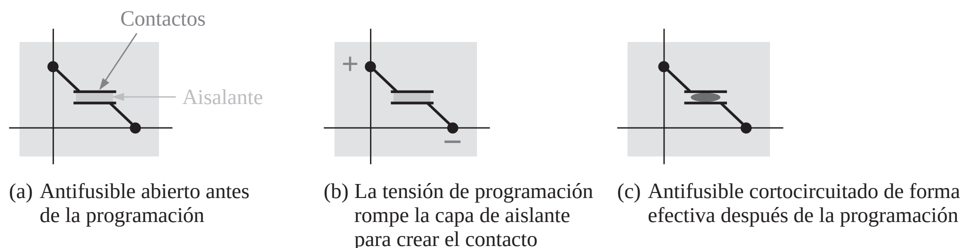


FIGURA 3.52 Conexión programable mediante antifusible.

Tecnología basada en EPROM. En determinados dispositivos lógicos programables, las conexiones programables son similares a las celdas de memoria de las ***EPROM*** (*Electrically Programmable Read–Only Memory*, memoria de sólo lectura eléctricamente programable). Este tipo de PLD se programa empleando una herramienta especial conocida como programador de dispositivos. El dispositivo se inserta en el programador, el cual está conectado a una computadora que ejecuta el software de programación. La mayoría de los PLD basados en EPROM son dispositivos programables una sola vez (OTP). Sin embargo, aquellos con encapsulado de ventana pueden borrarse utilizando luz ultravioleta (UV) y volverse a programar utilizando una herramienta estándar de programación de dispositivos PLD. La tecnología EPROM emplea un tipo especial de transistor MOS, conocido como transistor de puerta flotante, como conexión programable. El dispositivo de puerta flotante usa un proceso denominado tunelización de Fowler–Nordheim para colocar electrones en la estructura de puerta flotante.

En una matriz AND programable, el transistor de puerta flotante actúa como un interruptor que permite conectar una fila bien a un nivel ALTO o a un nivel BAJO, dependiendo de la variable de entrada. Para las variables de entrada que no se utilizan, el transistor se programa de manera que esté permanentemente *desactivado* (abierto). La Figura 3.53 muestra una puerta AND en una matriz simple. La variable A controla el estado del transistor de la primera columna y la variable B controla el transistor de la tercera columna. Cuando un transistor está desactivado (no conduce), es decir, se comporta como un interruptor abierto, la línea de entrada a la puerta AND está a un nivel $+V$ (ALTO). Cuando un transistor está activado (*on*), es decir, se comporta como un interruptor cerrado, la línea de entrada se conecta a tierra (nivel BAJO). Si la variable A o B está a 0 (nivel BAJO), el transistor está activado manteniendo la línea de entrada a la puerta AND a nivel BAJO. Si la variable A o B está a 1 (nivel ALTO), el transistor no está *activado* (*off*) manteniendo la línea de entrada a la puerta AND a nivel ALTO.

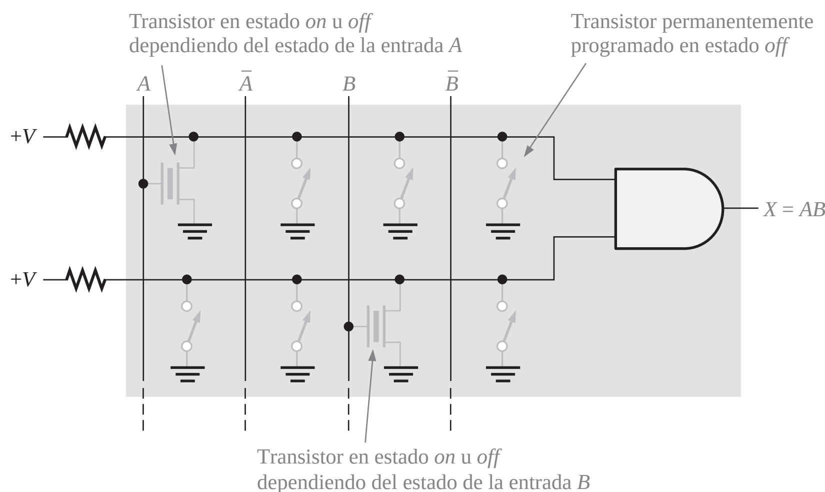
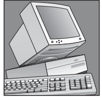


FIGURA 3.53 Una matriz AND con tecnología EPROM. Por simplicidad, sólo se muestra una puerta en la matriz.

Tecnología basada en EEPROM. La tecnología EEPROM, (*Electrically Erasable Programmable Read–Only Memory*) es similar a la EPROM porque también utiliza un tipo de transistor de puerta flotante en celdas E^2CMOS. La diferencia es que las EEPROM se pueden borrar y volver a programar eléctricamente sin tener que usar luz UV o herramientas especiales. Un dispositivo E^2CMOS puede programarse después de haberse montado en una tarjeta de circuito impreso y muchos pueden reprogramarse mientras que operan dentro de un sistema. Esto se denomina programación dentro del sistema (**ISP**, *In–System Programming*). La Figura 3.53 también puede verse como un ejemplo para representar una matriz AND con tecnología EEPROM.

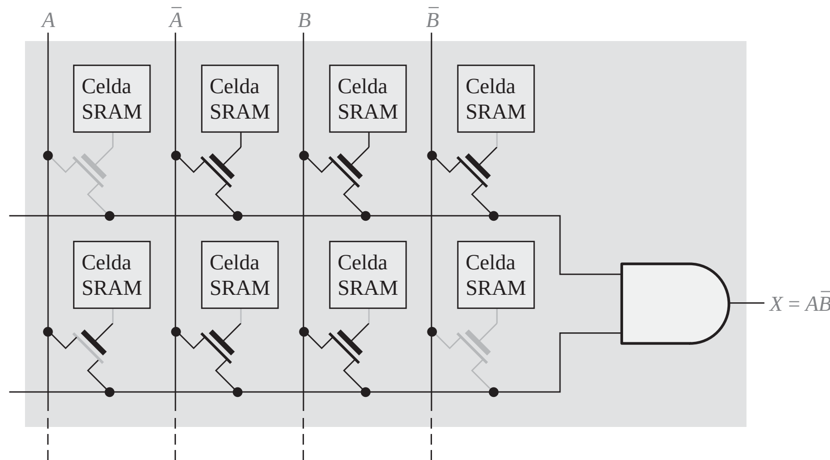


NOTAS INFORMÁTICAS

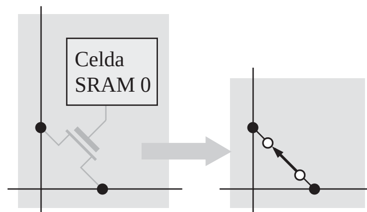
La mayoría de los diseños en el nivel de sistema incorporan diversos dispositivos como memorias RAM, ROM, controladores y procesadores que se interconectan mediante una gran cantidad de dispositivos lógicos de propósito general que, a menudo, se dice que son el "pegamento" lógico. Los PLD están reemplazando a muchos de los dispositivos "pegamento" SSI y MSI. El uso de los PLD proporciona una reducción en el número de encapsulados, por ejemplo, en los sistemas de memoria de las computadoras, los PLD pueden emplearse para la decodificación de direcciones de memoria y para generar señales de escritura en memoria, así como otras funciones.

Tecnología basada en SRAM. Muchas FPGA y algunos CPLD utilizan una tecnología de proceso similar a la que emplean las memorias ***SRAM*** (*Static Random Access Memory*, memoria estática de acceso aleatorio). El concepto fundamental de las matrices lógicas programables basadas en SRAM se ilustra en la Figura 3.54(a). Se emplea una celda de memoria tipo SRAM para activar o desactivar un transistor con el fin de conectar o desconectar las filas y columnas. Por ejemplo, cuando la celda de memoria contiene un 1 (en gris), el transistor se *activa* (*on)* y conecta las líneas de fila y columna asociadas, como se muestra en la parte (b) de la figura. Si la celda de memoria contiene un 0 (en negro), el transistor se *desactiva* (*off*) y no se establece ninguna conexión entre las líneas, como se muestra en la parte (c).

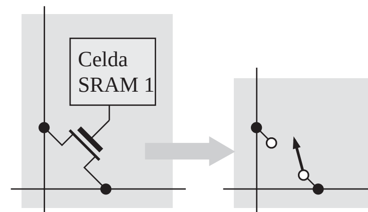
La tecnología basada en SRAM es diferente de las otras tecnologías de proceso vistas hasta ahora en el sentido de que es una tecnología volátil. Esto quiere decir que una celda SRAM no conserva los datos cuando se desconecta la alimentación. Los datos de la programación deben cargarse en una memoria no volátil y, cuando se conecta la alimentación, los datos almacenados en memoria reprograman el PLD basado en SRAM.



(a) Matriz programable basada en SRAM



(b) Transistor *on*



(c) Transistor *off*

FIGURA 3.54 Concepto básico de una matriz AND con tecnología basada en SRAM.

Las tecnologías de proceso basadas en fusible, antifusible, EPROM y EEPROM son no volátiles, por lo que conservan su programación cuando se desconecta la alimentación. Un fusible queda permanentemente abierto, un antifusible queda permanentemente cerrado y los transistores de puerta flotante empleados en las matrices basadas en EPROM y EEPROM pueden conservar indefinidamente sus estados *on* u *off*.

Programación de dispositivos

En el Capítulo 1 se ha presentado el concepto general de programación y también hemos visto cómo pueden realizarse interconexiones en una matriz sencilla abriendo y cerrando las conexiones programables. Los SPLD, los CPLD y las FPGA se programan prácticamente de la misma forma. Los dispositivos que emplean una tecnología de proceso OTP, como las basadas en fusibles, antifusibles o EPROM, deben programarse empleando una herramienta hardware especial denominada *programador*. El programador se conecta a una computadora mediante un cable de interfaz estándar, como se muestra en la Figura 3.55. El software de desarrollo se instala en la computadora y el dispositivo se inserta en el zócalo del programador. La mayoría de los programadores disponen de adaptadores, como el que se muestra en la figura, que permiten insertar diferentes tipos de encapsulados.

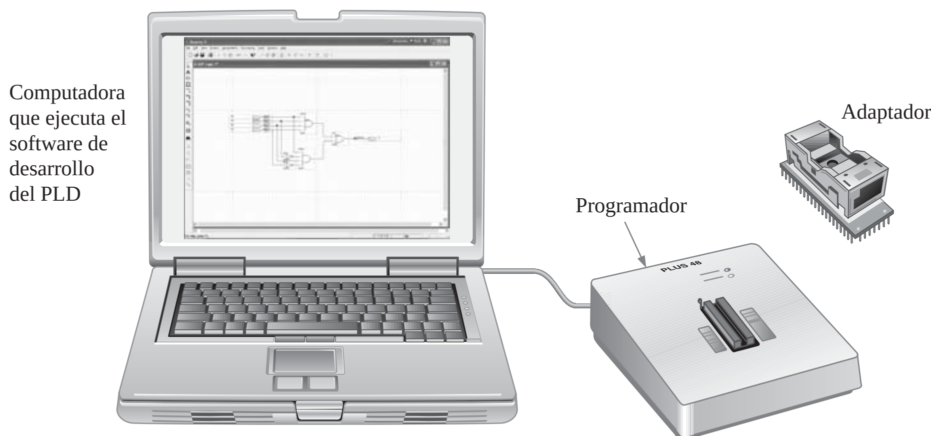


FIGURA 3.55 Configuración para programar un PLD con una herramienta de programación (programador).

Los dispositivos lógicos programables basados en EEPROM y SRAM son reprogramables y pueden reconfigurarse múltiples veces. Aunque para este tipo de dispositivo puede utilizarse un programador de dispositivos, generalmente, se programan inicialmente sobre una tarjeta de desarrollo de PLD, como se muestra en la Figura 3.56. Puede desarrollarse un diseño lógico utilizando esta técnica porque cualquier cambio necesario que surja durante el proceso de diseño podrá implementarse fácilmente simplemente reprogramando el PLD. Un PLD en el que se puede descargar un diseño lógico software se denomina ***dispositivo objetivo***. Además del dispositivo objetivo, normalmente, las tarjetas de desarrollo proporcionan circuitería adicional y conectores para poder establecer la interfaz con la computadora y otros circuitos periféricos. Además, la tarjeta de desarrollo dispone de puntos de prueba y de dispositivos de presentación que permiten observar el funcionamiento del dispositivo programado

Introducción del diseño. Como hemos visto en el Capítulo 1, la introducción del diseño consiste en programar el diseño lógico empleando el software de desarrollo. Las dos formas principales para introducir un diseño son mediante una interfaz de texto o mediante una interfaz gráfica (esquemáticos), por lo que habitual-

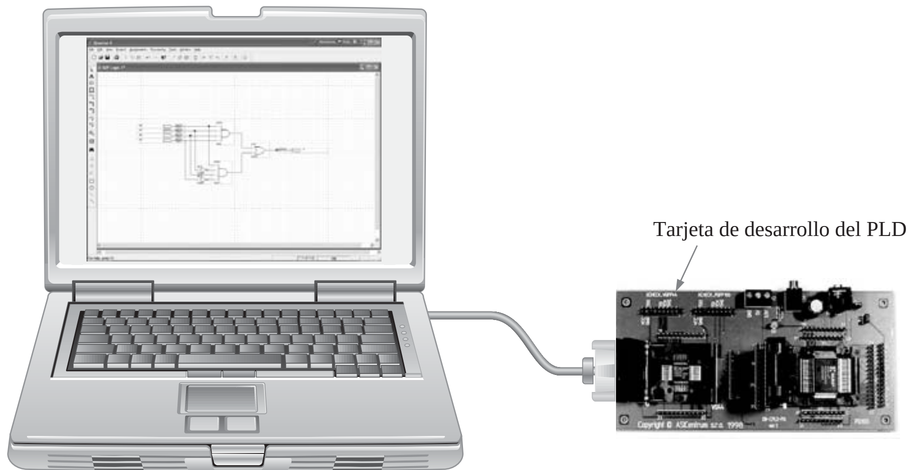


FIGURA 3.56 Configuración para programar dispositivos lógicos reprogramables.

mente los fabricantes de dispositivos lógicos programables proporcionan paquetes de software para sus dispositivos que permiten utilizar ambos métodos para la introducción de datos.

En la mayoría de los software de desarrollo, independientemente del fabricante, la **interfaz de texto** permite emplear dos o más lenguajes de desarrollo hardware (**HDL**, *Hardware Development Language*). Por ejemplo, todos los paquetes de software soportan los lenguajes estándar HDL del IEEE, VHDL y Verilog. Algunos paquetes de software también permiten el uso de ciertos lenguajes propietarios como ABEL, CUPL y AHDL.

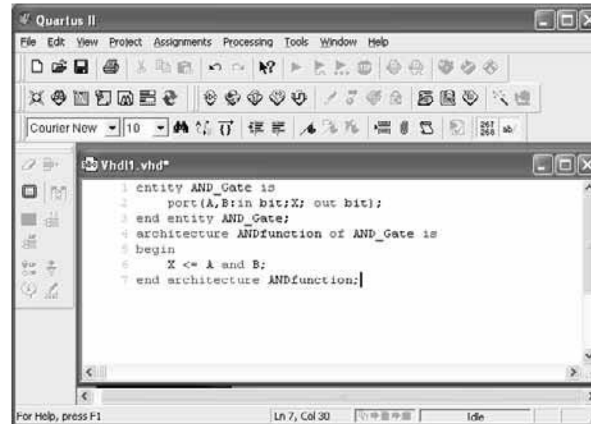
Con la **interfaz gráfica (esquemáticos),** los símbolos lógicos como por ejemplo, de las puertas AND y OR se colocan en la pantalla y se interconectan para formar el circuito deseado. Con este método, se emplean los símbolos lógicos familiares, aunque lo que hace realmente el software es convertir cada símbolo y las interconexiones en un archivo de texto que la computadora puede utilizar; el usuario no ve este proceso. En la Figura 3.57 se muestra un ejemplo de ambas interfaces. Por regla general, la interfaz gráfica se utiliza para los circuitos lógicos menos complejos y la interfaz de texto, aunque puede utilizarse para lógica sencilla, se usa para implementaciones más complejas.

Programación dentro del sistema (ISP)

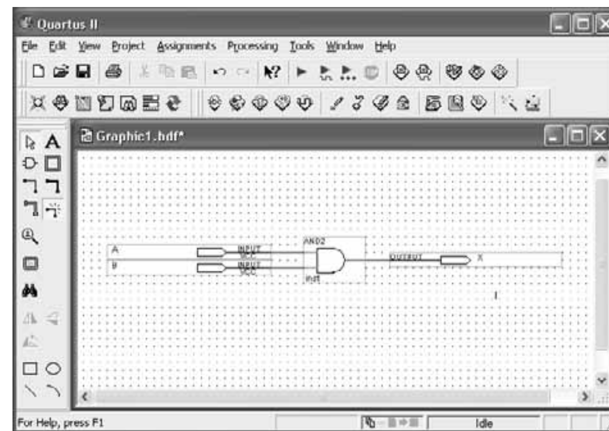
Ciertos CPLD y FPGA pueden programarse después de haberse incluido en la placa de circuito impreso (PCB) de un sistema. Después de haber desarrollado y probado por completo un diseño lógico en una tarjeta de desarrollo, se puede programar entonces un dispositivo "en blanco" que ya haya sido soldado sobre la placa del sistema en que tendrá que operar. También, si es necesario incorporar un cambio en el diseño, el dispositivo ya montado en la placa puede reconfigurarse para incluir las modificaciones al diseño.

En el caso de un diseño ya en producción, la programación de un dispositivo montado en la tarjeta del sistema minimiza la manipulación y elimina la necesidad de mantener stocks de dispositivos preprogramados. Esto también descarta la posibilidad de que se incluyan componentes erróneos en un producto. Los dispositivos no programados (en blanco) pueden mantenerse en el almacén y programarse a medida que se necesiten. Esto minimiza el capital que necesita un negocio para inventarios y mejora la calidad de los productos.

JTAG. El estándar establecido por el grupo *Joint Test Action Group* es el nombre normalmente utilizado para el estándar IEEE 1149.1. El estándar ***JTAG*** fue desarrollado para proporcionar un método sencillo, denominado análisis de contorno (*boundary scan*), para probar la funcionalidad de los dispositivos programables, así como para probar circuitos impresos con el fin de detectar conexiones malas (pines cortocircuitados, pines



(a) Entrada de texto VHDL



(b) Entrada gráfica equivalente (esquemático)

FIGURA 3.57 Ejemplos de introducción del diseño de una puerta AND.

abiertos, pistas cortadas, etc.). Más recientemente, JTAG se ha utilizado como una forma adecuada de configurar los dispositivos programables dentro del sistema. A medida que la demanda de productos actualizables sobre el terreno aumente, el uso de JTAG como una técnica apropiada de reprogramación de dispositivos CPLD y FPGA continuará creciendo.

Los dispositivos compatibles con JTAG disponen de hardware interno dedicado que interpreta las instrucciones y datos proporcionados por cuatro señales dedicadas. El estándar JTAG define estas señales como TDI (*Test Data In*), TDO (*Test Data Out*), TMS (*Test Mode Select*) y TCK (*Test Clock*). El hardware dedicado JTAG interpreta las instrucciones y los datos de las señales TDI y TMS, y envía los datos en la señal TDO. La señal TCK se utiliza para sincronizar el proceso. En la Figura 3.58 se muestra una tarjeta de circuito impreso compatible con JTAG.

Procesador integrado. Otra técnica para la programación dentro del sistema es el uso de una memoria y un microprocesador integrado. El procesador se integra dentro del sistema junto con el CPLD o la FPGA y el resto de la circuitería, y tiene como objetivo establecer la configuración dentro del sistema del dispositivo programable.

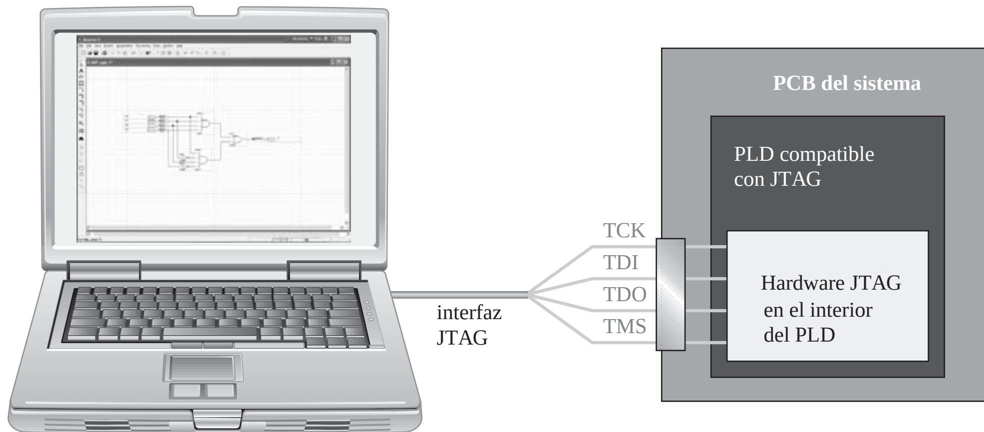


FIGURA 3.58 Ilustración simplificada de una programación dentro del sistema a través de una interfaz JTAG.

Como ya ha aprendido, los dispositivos basados en SRAM son volátiles y pierden los datos programados cuando se desconecta la alimentación. Por tanto, es necesario almacenar los datos de la programación en una PROM (*Programmable Read–Only Memory*, memoria programable de sólo lectura), que es no volátil. Cuando se conecta la alimentación, el procesador integrado toma el control para transferir los datos almacenados desde la PROM al CPLD o la FPGA.

Además, el procesador integrado en ocasiones se emplea para reconfigurar un dispositivo programable mientras que el sistema está en funcionamiento. En este caso, los cambios de diseño se realizan por software y los nuevos datos se cargan en un PROM sin perturbar el funcionamiento del sistema. El procesador controla la transferencia de los datos al dispositivo "sobre la marcha" durante un instante apropiado. En la Figura 3.59 se muestra un simple diagrama de bloques de un sistema lógico programable con procesador integrado.

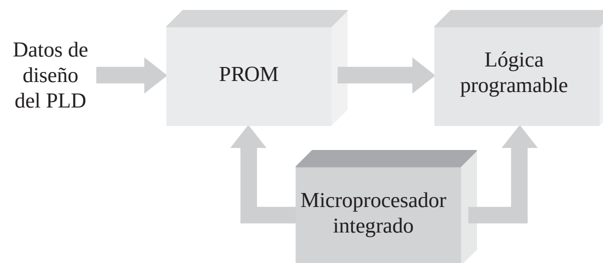


FIGURA 3.59 Diagrama de bloques simplificado de un PLD con una memoria y un procesador integrado.

REVISIÓN DE LA SECCIÓN 3.7

1. Enumere cinco tecnologías de proceso utilizadas para las conexiones programables en los dispositivos lógicos programables.
2. ¿Qué significa el término *volátil* en relación con los PLD y qué tecnologías de proceso son volátiles?
3. ¿Cuáles son los dos métodos disponibles para la introducción del diseño en la programación de los PLD y las FPGA?
4. Defina JTAG.

3.8 LÓGICA DE FUNCIÓN FIJA

Existen dos tecnologías de circuitos integrados digitales que se usan para implementar las puertas lógicas básicas: CMOS y TTL. Las operaciones lógicas NOT, AND, OR, NAND, NOR y OR–exclusiva son las mismas, independientemente de la tecnología de circuitos integrados que se utilice; es decir, una puerta AND tiene la misma función lógica se implemente con la tecnología CMOS o TTL.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Identificar las series más comunes CMOS y TTL. ■ Comparar las tecnologías CMOS y TTL en términos de tipos de dispositivos y parámetros de funcionamiento. ■ Definir el *retardo de propagación*. ■ Definir la *disipación de potencia*. ■ Definir el *fan–out*. ■ Definir el producto *velocidad–potencia*. ■ Interpretar la información básica de una hoja de características.

El término ***CMOS*** corresponde a *Complementary Metal–Oxide Semiconductor* (semiconductor metal–óxido complementario) y se implementa con un tipo de transistor de efecto de campo. ***TTL*** (*Transistor-Transistor Logic*, lógica transistor–transistor) y se implementa mediante transistores bipolares. Tenga en cuenta que CMOS y TTL sólo difieren en el tipo de componentes de circuito y los valores de los parámetros, y no en las operaciones lógicas básicas. Una puerta AND CMOS realiza la misma operación lógica que una puerta AND TTL. Esto también es cierto para todas las operaciones lógicas básicas restantes. La diferencia entre CMOS y TTL se encuentra en las características de funcionamiento, tal como la velocidad de conmutación (retardo de propagación), la disipación de potencia, la inmunidad al ruido y otros parámetros.

CMOS

Existe poco desacuerdo sobre cuál es la tecnología de circuitos, CMOS o TTL, más ampliamente utilizada. Parece que CMOS está comenzando a ser la tecnología dominante y podría reemplazar a la tecnología TTL en los CI de pequeña y mediana escala. Aunque TTL ha dominado durante muchos años, principalmente debido a sus altas velocidades de conmutación y a una enorme variedad de tipos de dispositivos, la tecnología CMOS siempre ha tenido la ventaja de ofrecer una mucho menor disipación de potencia, aunque dicho parámetro depende de la frecuencia. Las velocidades de conmutación de CMOS han mejorado extremadamente y ahora pueden competir con TTL, a la vez que la baja disipación de potencia y otros factores deseables se han mantenido a medida que la tecnología avanzaba.

Series CMOS. Las categorías de CMOS en términos de tensión de alimentación continua son la serie CMOS de 5 V, la serie CMOS de 3,3 V, la serie CMOS de 2,5 V y la serie CMOS de 1,8 V. Las series CMOS de más baja tensión son el resultado de un desarrollo más reciente y de un esfuerzo por reducir la disipación de potencia. Puesto que la disipación de potencia es proporcional al cuadrado de la tensión, una reducción de 5 V a 3,3 V, por ejemplo, disminuye la potencia en un 34%, sin que el resto de los factores varíen.

Dentro de cada categoría según la tensión de alimentación, hay disponibles varias series de puertas lógicas CMOS. Estas series pertenecientes a la familia CMOS difieren en sus características de funcionamiento y se designan mediante los prefijos 74 ó 54, seguidos de una letra o letras que indican la serie y, a continuación, un número que indica el tipo de dispositivo lógico. El prefijo 74 indica que se trata de un dispositivo comercial de propósito general, y el prefijo 54 indica que es un dispositivo militar para aplicaciones en entornos más exigentes. En este libro sólo haremos referencia a los dispositivos que llevan el prefijo 74. La serie básica CMOS de 5 V y sus denominaciones son las siguientes:

- 74HC y 74HCT. CMOS de alta velocidad (la “T” indica compatibilidad TTL)
- 74AC y 74ACT. CMOS avanzada
- 74AHC y 74AHCT. CMOS de alta velocidad avanzada

La serie básica CMOS de 3,3 V y sus denominaciones son las siguientes:

- 74LV. CMOS de baja tensión
- 74LVC. CMOS de baja tensión.
- 74ALVC. CMOS de baja tensión avanzada.

Además de la serie 74, todavía existe la serie 4000, que es una tecnología CMOS más antigua y de baja velocidad, aunque su uso está limitado. Además de las series CMOS "puras" existen series que combinan ambas tecnologías, CMOS y TTL, que se denominan BiCMOS. La serie básica BiCMOS y sus denominaciones son las siguientes:

- 74BCT. BiCMOS
- 74ABT. BiCMOS avanzada.
- 74LVT. BiCMOS de baja tensión.
- 74ALB. BiCMOS de baja tensión.

TTL

La tecnología TTL ha sido y es todavía una tecnología de circuitos integrados digitales muy popular. Una ventaja de esta tecnología es que no es sensible a las descargas electrostáticas como lo es la tecnología CMOS y, por tanto, es más práctica en la realización de experimentos de laboratorio y la elaboración de prototipos, ya que no es necesario preocuparse por los problemas de manipulación.

Series TTL. Al igual que con la tecnología CMOS, hay disponibles varias series de puertas lógicas TTL, las cuales operan todas ellas con 5 V de alimentación de continua. Estas series pertenecientes a la familia TTL difieren en sus características de funcionamiento y se denominan mediante los prefijos 74 ó 54 seguidos por una letra o letras que indican la serie y un número que indica el tipo de dispositivo lógico de la serie. Un circuito integrado TTL puede distinguirse de un circuito integrado CMOS por las letras que siguen a los prefijos 74 y 54.

Las series básicas TTL y sus denominaciones son las siguientes:

- 74: TTL estándar (sin letra).
- 74S: TTL Schottky.
- 74AS: TTL Schottky avanzada.
- 74LS: TTL Schottky de baja potencia.
- 74ALS: TTL Schottky de baja potencia avanzada.
- 74F: TTL rápida.

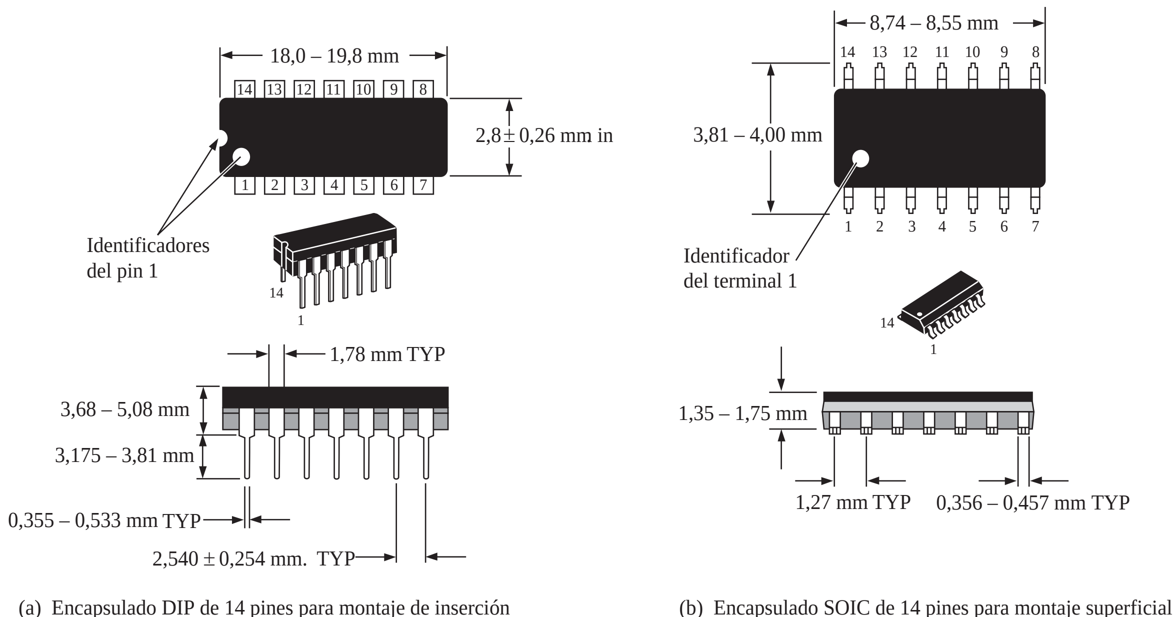
Tipos de puertas lógicas de función fija

Todas las operaciones lógicas básicas: NOT, AND, OR, NAND, NOR, OR–exclusiva (XOR) y NOR–exclusiva (XNOR) están disponibles en las tecnologías CMOS y TTL. También están disponibles puertas con salida búfer para excitar cargas que requieran altas corrientes. Los tipos de configuraciones de puerta normalmente disponibles en los circuitos integrados se identifican mediante los dos o tres dígitos finales de la designación de la serie. Por ejemplo, 74LS04 es un circuito integrado inversor séxtuple **Schottky** de baja potencia. Algunas de las configuraciones de puertas lógicas habituales y sus dígitos de identificación estándar son los siguientes:

- Cuádruple NAND de dos entradas: **00**

- Cuádruple NOR de dos entradas: **02**
- Inversor séxtuple: **04**
- Cuádruple AND de dos entradas: **08**
- Triple NAND de tres entradas: **10**
- Triple AND de tres entradas: **11**
- Doble NAND de cuatro entradas: **20**
- Doble AND de dos entradas: **21**
- Triple NOR de tres entradas: **27**
- NAND de ocho entradas: **30**
- Cuádruple OR de dos entradas: **32**
- Cuádruple XOR: **86**
- Cuádruple XNOR: **266**

Encapsulados de circuitos integrados. Todos los CMOS de la serie 74 son compatibles en su patillaje con respecto a los mismos tipos de dispositivos en tecnología TTL. Esto quiere decir que un circuito integrado digital CMOS, como el 74AHC00 (cuádruple NAND de 2 entradas), que contiene cuatro puertas NAND de 2 entradas en un único circuito integrado, tiene exactamente los mismos números de pin para cada entrada y salida que el correspondiente dispositivo TTL. En la Figura 3.60 se muestra los encapsulados típicos de circuitos integrados, el encapsulado DIP (*Dual In–line Package*) para montaje de inserción y el encapsulado SOIC (*Small–Outline Integrated Circuit*) para montaje superficial. En algunos casos, hay disponibles algunos otros tipos de encapsulados. Aunque no se muestra a escala, el encapsulado SOIC es significativamente más pequeño que el encapsulado DIP. Los diagramas de configuración de los pines para la mayor parte de los dispositivos lógicos enumerados anteriormente se muestran en la Figura 3.61.



(a) Encapsulado DIP de 14 pines para montaje de inserción

(b) Encapsulado SOIC de 14 pines para montaje superficial

FIGURA 3.60 Encapsulados típicos DIP y SOIC con sus dimensiones básicas y la numeración de los pines.

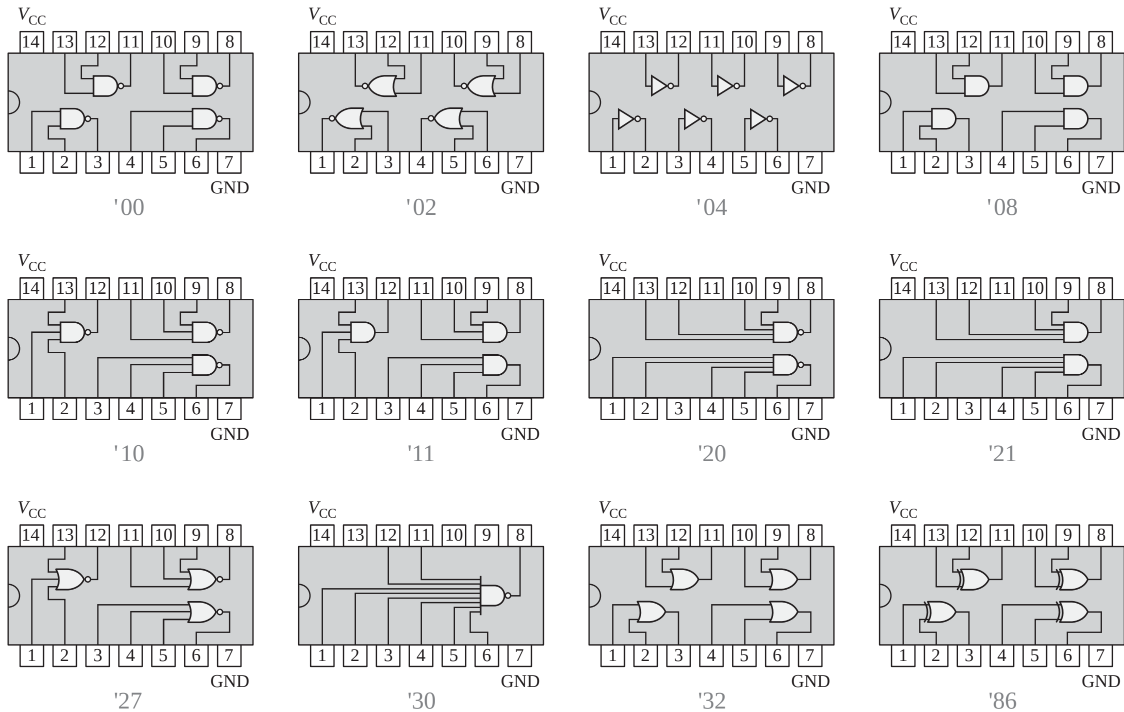


FIGURA 3.61 Diagramas de configuración de los pines para algunas de las configuraciones de puertas integradas de función fija más comunes.

Lógica monopuerta. Hay disponible una selección limitada de puertas CMOS en encapsulados monopuerta. Con una única puerta por chip, esta serie se vende en pequeños encapsulados de cinco pines que sirven para llevar a cabo las modificaciones de último momento para incluir determinados circuitos lógicos en zonas de alta densidad, cuando el espacio disponible es limitado.

Símbolos lógicos. Los símbolos lógicos para los circuitos integrados de función fija utilizan los símbolos de puerta estándares y presentan el número de puertas en el encapsulado del circuito integrado y los números de pines asociados a cada puerta, así como los números de pines para V_{CC} y masa. En la Figura 3.62 se muestra un ejemplo para el inversor séxtuple y para la puerta cuádruple NAND de 2 entradas. Se muestran el diagrama lógico y el símbolo rectangular. Independientemente de la familia lógica, todos los dispositivos con el mismo sufijo tienen pines compatibles; es decir, tienen la misma disposición de números de pines. Por ejemplo, los dispositivos 7400, 74S00, 74LS00, 74ALS00, 74F00, 74HC00 y 74AHC00 son todos ellos circuitos integrados cuádruples NAND de 2 entradas compatibles en cuanto a sus pines.

Características y parámetros de funcionamiento

▲ *La lógica de alta velocidad presenta un tiempo de retardo de propagación corto.*

Existen varios puntos que definen el funcionamiento de un circuito lógico. Las características de funcionamiento son: la velocidad de conmutación medida en términos del retardo de propagación, la disipación de potencia, el fan–out o capacidad de excitación, el producto velocidad–potencia, la tensión de alimentación continua y los niveles lógicos de entrada/salida.

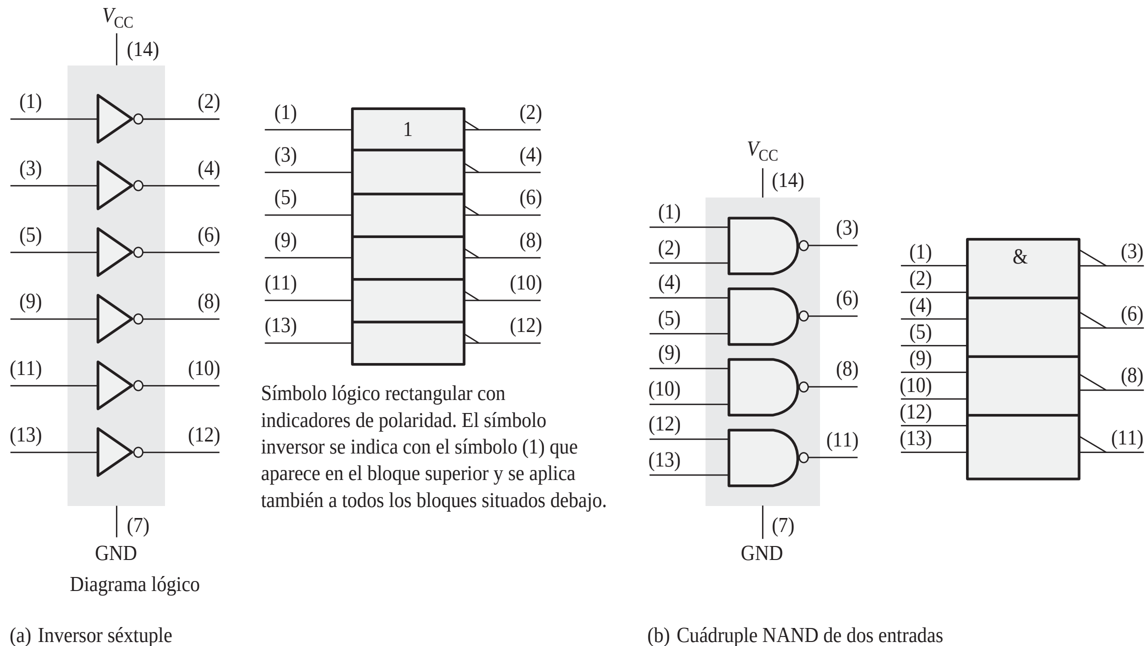


FIGURA 3.62 Símbolos lógicos para el inversor séxtuple (sufijo 04) y la puerta NAND cuádruple de 2 entradas (sufijo 00). El símbolo se aplica al mismo dispositivo en cualquier serie CMOS o TTL.

Tiempo de retardo de propagación. Este parámetro limita la frecuencia o velocidad de conmutación a la que un circuito lógico puede operar. Cuando se aplican a los circuitos lógicos, los términos *baja velocidad* y *alta velocidad* hacen referencia al retardo de propagación. Cuanto menor sea el tiempo de propagación, mayor será la velocidad del circuito y mayor será la frecuencia a la que puede operar.

El **tiempo de retardo de propagación**, t_P, de una puerta lógica es el intervalo de tiempo entre la aplicación de un impulso de entrada y la aparición del impulso de salida resultante. Existen dos medidas diferentes del tiempo de retardo de propagación asociado con una puerta lógica, que se aplican a todos los tipos de puertas básicas:

- t_{PHL}: es el tiempo entre un punto de referencia especificado en el impulso de entrada y el correspondiente punto de referencia en el impulso de salida, cuando la salida cambia del nivel ALTO (H) al nivel BAJO (L).
- t_{PLH}: es el tiempo entre un punto de referencia especificado en el impulso de entrada y el correspondiente punto de referencia en el impulso de salida, cuando la salida cambia del nivel BAJO (L) al nivel ALTO (H).

EJEMPLO 3.22

Determinar los tiempos de propagación en el inversor de la Figura 3.63(a).

Solución Los tiempos de propagación, t_{PHL} y t_{PLH}, se indican en la parte (b) de la figura. En este caso, los retrasos se miden entre los puntos de pendiente del 50% en los correspondientes flancos de los impulsos de entrada y salida. Los valores de t_{PHL} y t_{PLH} no necesariamente son iguales pero, en la mayoría de los casos, sí lo son.

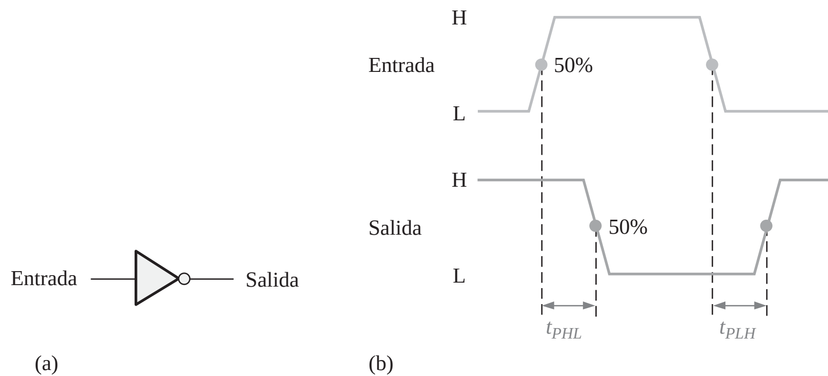


FIGURA 3.63

Problema relacionado Una puerta lógica tiene una valor máximo especificado de 10 ns para t_{PHL} y t_{PLH}. Para otros tipos de puertas este valor es de 4 ns. ¿Cuál es la puerta que puede trabajar a la frecuencia mayor?

En las puertas TTL de la serie estándar, el retardo de propagación típico es de 11 ns y para las puertas de la serie F es de 3,3 ns. Para las puertas CMOS de la serie HCT, el retardo de propagación es 7 ns, para la serie AC es de 5 ns y para la serie ALVC es de 3 ns. Todos los valores especificados dependen de determinadas condiciones de operación, tal y como se establece en las hojas de características.

Tensión de alimentación continua (V_{CC}). La tensión de alimentación continua típica para CMOS puede ser 5 V; 3,3 V, 2,5 V o 1,8 V, dependiendo de la categoría. Una ventaja de CMOS es que las tensiones de alimentación pueden variar en un rango más amplio que los dispositivos TTL. Los CMOS de 5 V pueden tolerar variaciones de alimentación desde los 2 V a 6 V y aún así funcionarán adecuadamente, aunque el retardo de propagación y la disipación de potencia se vean significativamente afectadas. Los dispositivos CMOS de 3,3 V pueden operar con tensiones de alimentación desde 2 V hasta 3,6 V. La tensión de alimentación continua típica para dispositivos TTL es 5,0 V con un mínimo de 4,5 V y un máximo de 5,5 V.

Disipación de potencia. La **disipación de potencia**, P_D, de una puerta lógica es el producto de la tensión de alimentación continua y de la corriente media de alimentación. Normalmente, la corriente de alimentación cuando la salida de la puerta está a nivel BAJO es mayor que cuando la salida de la puerta está a nivel ALTO. Generalmente, las hojas de características del fabricante especifican la corriente de alimentación para el estado de salida BAJO como I_{CCL} y para el estado ALTO como I_{CCH}. La corriente media de alimentación se determina en función de un ciclo de trabajo del 50% (nivel de salida BAJO la mitad del tiempo y la otra mitad nivel de salida ALTO), por tanto la disipación de potencia media de una puerta lógica es

Ecuación 3.2

$$P_D = V_{CC}\left(\frac{I_{CCH} + I_{CCL}}{2}\right)$$

Las puertas de la serie CMOS tienen disipaciones de potencia muy bajas en comparación con las series TTL. Sin embargo, la disipación de potencia en los dispositivos CMOS depende de la frecuencia de funcionamiento. Para una frecuencia cero, la potencia está normalmente en el rango de los microvatios por puerta, y en la frecuencia máxima de funcionamiento puede estar en el rango de los milivatios; por tanto, algunas veces la potencia se especifica para una frecuencia determinada. Por ejemplo, la serie HC tiene una potencia de 2,75 μW/puerta para una frecuencia igual a 0 Hz y de 600 μW/puerta para 1 MHz.

La disipación de potencia para los dispositivos TTL es independiente de la frecuencia. Por ejemplo, la serie ALS disipa 1,4 mW/puerta, independientemente de la frecuencia y la serie F disipa 6 mW/puerta.

Niveles lógicos de entrada y salida. V_{IL} es la tensión del nivel de entrada BAJO para una puerta lógica y V_{IH} es la tensión de entrada del nivel ALTO. Los dispositivos CMOS de 5 V aceptan una tensión máxima de 1,5 V para V_{IL} y una tensión mínima de 3,5 V para V_{IH}. Los dispositivos TTL aceptan una tensión máxima de 0,8 V para V_{IL} y una tensión mínima de 2 V para V_{IH}.

V_{OL} es la tensión de salida para el nivel BAJO y V_{OH} es la tensión de salida para el nivel ALTO. Para los dispositivos CMOS de 5 V, el valor máximo de V_{OL} es de 0,33 V y el valor mínimo para V_{OH} es de 4,4 V. Para los dispositivos TTL, el valor máximo V_{OL} es de 0,4 V y el mínimo V_{OH} es de 2,4 V. Todos los valores dependen de las condiciones de operación, tal y como se especifica en la hoja de características.

Producto velocidad-potencia (SPP). El parámetro **SPP** (*Speed–Power Product*) puede utilizarse como una medida del funcionamiento de un circuito lógico que tiene en cuenta el retardo de propagación y la disipación de potencia. Es especialmente útil para comparar las distintas series de puertas lógicas de las familias CMOS o TTL o para comparar una puerta CMOS con una puerta TTL.

El producto SPP de un circuito lógico es igual al producto del retardo de propagación por la disipación de potencia, y se expresa en julios (J), que es una unidad de energía. La fórmula es

Ecuación 3.3

$$SPP = t_p P_D$$

EJEMPLO 3.23

Una determinada puerta tiene un retardo de propagación de 5 ns, I_{CCH} = 1 mA e I_{CCL} = 2,5 mA, con una tensión de alimentación continua de 5 V. Determinar el producto velocidad–potencia.

Solución

$$P_D = V_{CC}\left(\frac{I_{CCH} + I_{CCL}}{2}\right) = 5\,\text{V}\left(\frac{1\,\text{mA} + 2,5\,\text{mA}}{2}\right) = 5\,\text{V}(1,75\,\text{mA}) = 8,75\,\text{W}$$

$$SPP = (5\,\text{ns})(8,75\,\text{mW}) = \mathbf{43,75\,pJ}$$

Problema relacionado Si el retardo de propagación de una puerta es 15 ns y su *SPP* es igual a 150 pJ, ¿cuál es su disipación de potencia media?

Fan-out y carga. El **fan–out** de una puerta lógica es el número máximo de entradas de la familia de circuitos integrados de la misma serie que la puerta puede excitar, manteniendo a la vez los niveles de salida dentro de los límites especificados. El fan–out es un parámetro importante sólo en la tecnología TTL. Dado que con los circuitos CMOS se asocian impedancias muy altas, el fan–out es muy alto, aunque depende de la frecuencia debido a los efectos capacitivos.

El fan–out se especifica en términos de ***cargas unidad***. Una carga unidad para una puerta lógica es igual a una entrada de un circuito similar. Por ejemplo, una carga unidad para una puerta NAND 74LS00 es igual a *una* entrada a una puerta lógica en la serie 74LS (no necesariamente una puerta NAND). Puesto que la corriente para una entrada a nivel BAJO (I_{IL}) de una puerta 74LS00 es de 0,4 mA y la corriente que una salida a nivel BAJO (I_{IL}) puede aceptar es de 8,0 mA, el número de cargas unidad que una puerta 74LS00 puede excitar en el estado BAJO es

$$\text{Cargas unidad} = \frac{I_{OL}}{I_{IL}} = \frac{8,0\,\text{mA}}{0,4\,\text{mA}} = 20$$

La Figura 3.64 muestra puertas lógicas LS que excitan una serie de puertas con la misma tecnología de circuitos, donde el número de puertas depende de la tecnología de circuitos particular. Por ejemplo, como hemos visto, el número máximo de entradas de puerta (cargas unidad) que una puerta TTL de la serie 74LS puede excitar es 20.

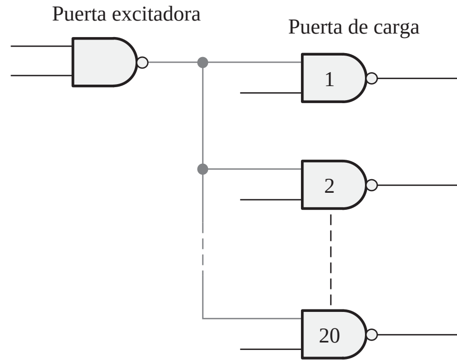


FIGURA 3.64 La salida de la puerta NAND LS TTL admite como carga máxima la entrada de 20 puertas LS TTL.

QUAD 2-INPUT NAND GATE

• ESD > 3500 Volts

SN54/74LS00

QUAD 2-INPUT NAND GATE
LOW POWER SCHOTTKY

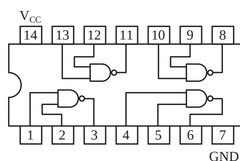
J SUFFIX
CERAMIC
CASE 632-08

N SUFFIX
PLASTIC
CASE 646-06

D SUFFIX
SOIC
CASE 751A-02

ORDERING INFORMATION

SN54LSXXJ	Ceramic
SN74LSXXN	Plastic
SN74LSXXD	SOIC



SN54/74LS00

DC CHARACTERISTICS OVER OPERATING TEMPERATURE RANGE (unless otherwise specified)

Symbol	Parameter		Limits Min	Typ	Max	Unit	Test Conditions
V_{IH}	Input HIGH Voltage		2.0			V	Guaranteed Input HIGH Voltage for All Inputs
V_{IL}	Input LOW Voltage	54			0.7	V	Guaranteed Input LOW Voltage for All Inputs
		74			0.8		
V_{IK}	Input Clamp Diode Voltage			–0.65	–1.5	V	V_{CC} = MIN, I_{IN} = –18 mA
V_{OH}	Ouput HIGH Voltage	54	2.5	3.5		V	V_{CC} = MIN, I_{OH} = MAX, V_{IN} = V_{IH} or V_{IL} per Truth Table
		74	2.7	3.5		V	
V_{OL}	Ouput LOW Voltage	54, 74		0.25	0.4	V	I_{OL} = 4.0 mA; V_{CC} = V_{CC} MIN, V_{IN} = V_{IL} or V_{IH} per Truth Table
		74		0.35	0.5	V	I_{OL} = 8.0 mA
I_{IH}	Input HIGH Current				20	μA	V_{CC} = MAX, V_{IN} = 2.7 V
					0.1	mA	V_{CC} = MAX, V_{IN} = 7.0 V
I_{IL}	Input LOW Current				–0.4	mA	V_{CC} = MAX, I_N = 0.4 V
I_{OS}	Short Circuit Current (Note 1)		–20		–100	mA	V_{CC} = MAX
I_{CC}	Power Supply Current Total, Output HIGH				1.6	mA	V_{CC} = MAX
	Total, Output LOW				4.4		

NOTE 1: Not more than one output should be shorted at a time, nor for more than 1 second.

AC CHARACTERISTICS (T_A = 25 C)

Symbol	Parameter	Limits Min	Typ	Max	Unit	Test Conditions
t_{PLH}	Turn-Off Delay, Input to Output		9.0	15	ns	V_{CC} = 5.0 V, C_L = 15 pF
t_{PHL}	Turn-On Delay, Input to Output		10	15	ns	

GUARANTEED OPERATING RANGES

Symbol	Parameter		Min	Typ	Max	Unit
V_{CC}	Supply Voltage	54	4.5	5.0	5.5	V
		74	4.75	5.0	5.25	
T_A	Operating Ambient Temperature Range	54	–55	25	125	C
		74	0	25	70	
I_{OH}	Output Current — High	54, 74			–0.4	mA
I_{OL}	Output Current — Low	54			4.0	mA
		74			8.0	

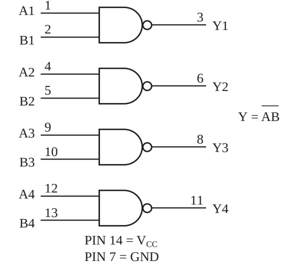
FIGURA 3.65 Parte de una hoja de características de un 74LS00.

Quad 2-Input NAND Gate **High-Performance Silicon–Gate CMOS**

The MC54/74HC00A is identical in pinout to the LS00. The device inputs are compatible with Standard CMOS outputs; with pullup resistors, they are compatible with LSTTL outputs.

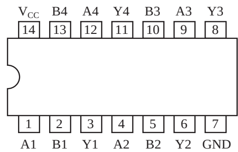
- Output Drive Capability: 10 LSTTLLoads
- Outputs Directly Interface to CMOS, NMOS andTTL
- OperatingVoltage Range: 2 to 6 V
- Low Input Current: 1μA
- High Noise Immunity Characteristic of CMOS Devices
- In ComplianceWith the JEDEC Standard No. 7A Requirements
- Chip Complexity: 32 FETs or 8 Equivalent Gates

LOGIC DIAGRAM



PIN 14 = V_{CC}
PIN 7 = GND

Pinout: 14–Load Packages (Top View)



MC54/74HC00A

J SUFFIX
CERAMIC PACKAGE
CASE 632-08

N SUFFIX
PLASTIC PACKAGE
CASE 646-06

D SUFFIX
SOIC PACKAGE
CASE 751A-03

DT SUFFIX
TSSOP PACKAGE
CASE 948G-01

ORDERING INFORMATION

MC54HCXXAJ	Ceramic
MC74HCXXAN	Plastic
MC74HCXXAD	SOIC
MC74HCXXADT	TSSOP

FUNCTION TABLE

Inputs		Output
A	B	Y
L	L	H
L	H	H
H	L	H
H	H	L

MAXIMUM RATINGS*

Symbol	Parameter	Value	Unit
V_{CC}	DC Supply Voltage (Referenced to GND)	- 0.5 to + 7.0	V
V_{in}	DC Input Voltage (Referenced to GND)	−0.5 to V_{CC} + 0.5	V
V_{out}	DC Output Voltage (Referenced to GND)	−0.5 to V_{CC} + 0.5	V
I_{in}	DC Input Current, per Pin	± 20	mA
I_{out}	DC Output Current, per Pin	± 25	mA
I_{CC}	DC Supply Current, V_{CC} and GND Pins	± 50	mA
P_D	Power Dissipation in Still Air, Plastic or Ceramic DIP	750	mW
	SOIC Package	500	
	TSSOP Package	450	
T_{stg}	Storage Temperature	– 65 to + 150	°C
T_L	Lead Temperature, 1 mm from Case for 10 Seconds		°C
	Plastic DIP, SOIC or TSSOP Package	260	
	Ceramic DIP	300	

* Maximum Ratings are those values beyond which damage to the device may occur.
Functional operation should be restricted to the Recommended Operating Conditions.
Derating Ñ Plastic DIP: – 10 mW/°C from 65° to 125° C
Ceramic DIP: – 10 mW/°C from 100° to 125° C
SOIC Package: – 7 mW/°C from 65° to 125° C
TSSOP Package: – 6.1 mW/°C from 65° to 125° C

RECOMMENDED OPERATING CONDITIONS

Symbol	Parameter	Min	Max	Unit
V_{CC}	DC Supply Voltage (Referenced to GND)	2.0	6.0	V
V_{in}, V_{out}	DC Input Voltage, Output Voltage (Referenced to GND)	0	V_{CC}	V
T_A	Operating Temperature, All Package Types	– 55	+125	°C
t_r, t_f	Input Rise and Fall Time V_{CC} = 2.0 V	0	1000	ns
	V_{CC} = 4.5 V	0	500	
	V_{CC} = 6.0 V	0	400	

DC CHARACTERISTICS (Voltages Referenced to GND) MC54/74HC00A

Symbol	Parameter	Condition	V_{CC} V	Guaranteed Limit –55 to 25°C	≤85°C	≤125°C	Unit
V_{IH}	Minimum High-Level Input Voltage	V_{out} = 0.1V or V_{CC} – 0.1V	2.0	1.50	1.50	1.50	V
		$\|I_{out}\| \leq 20\mu A$	3.0	2.10	2.10	2.10	
			4.5	3.15	3.15	3.15	
			6.0	4.20	4.20	4.20	
V_{IL}	Maximum Low-Level Input Voltage	V_{out} = 0.1V or V_{CC} – 0.1V	2.0	0.50	0.50	0.50	V
		$\|I_{out}\| \leq 20\mu A$	3.0	0.90	0.90	0.90	
			4.5	1.35	1.35	1.35	
			6.0	1.80	1.80	1.80	
V_{OH}	Minimum High-Level Output Voltage	$V_{in} = V_{IH}$ or V_{IL}	2.0	1.9	1.9	1.9	V
		$\|I_{out}\| \leq 20\mu A$	4.5	4.4	4.4	4.4	
			6.0	5.9	5.9	5.9	
		$V_{in} = V_{IH}$ or V_{IL} $\|I_{out}\| \leq 2.4mA$	3.0	2.48	2.34	2.20	
		$\|I_{out}\| \leq 4.0mA$	4.5	3.98	3.84	3.70	
		$\|I_{out}\| \leq 5.2mA$	6.0	5.48	5.34	5.20	
V_{OL}	Maximum Low-Level Output Voltage	$V_{in} = V_{IH}$ or V_{IL}	2.0	0.1	0.1	0.1	V
		$\|I_{out}\| \leq 20\mu A$	4.5	0.1	0.1	0.1	
			6.0	0.1	0.1	0.1	
		$V_{in} = V_{IH}$ or V_{IL} $\|I_{out}\| \leq 2.4mA$	3.0	0.26	0.33	0.40	
		$\|I_{out}\| \leq 4.0mA$	4.5	0.26	0.33	0.40	
		$\|I_{out}\| \leq 5.2mA$	6.0	0.26	0.33	0.40	
I_{in}	Maximum Input Leakage Current	$V_{in} = V_{CC}$ or GND	6.0	±0.1	±1.0	±1.0	μA
I_{CC}	Maximum Quiescent Supply Current (per Package)	$V_{in} = V_{CC}$ or GND, $I_{out} = 0\mu A$	6.0	1.0	10	40	μA

AC CHARACTERISTICS (C_L = 50 pF, Input $t_r = t_f$ = 6 ns)

Symbol	Parameter	V_{CC} V	Guaranteed Limit –55 to 25°C	≤85°C	≤125°C	Unit
t_{PLH}, t_{PHL}	Maximum Propagation Delay, Input A or B to Output Y	2.0	75	95	110	ns
		3.0	30	40	55	
		4.5	15	19	22	
		6.0	13	16	19	
t_{TLH}, t_{THL}	Maximum Output Transition Time, Any Output	2.0	75	95	110	ns
		3.0	27	32	36	
		4.5	15	19	22	
		6.0	13	16	19	
C_{in}	Maximum Input Capacitance		10	10	10	pF

		Typical @ 25°C, V_{CC} = 5.0 V, V_{EE} = 0 V	
C_{PD}	Power Dissipation Capacitance (Per Buffer)	22	pF

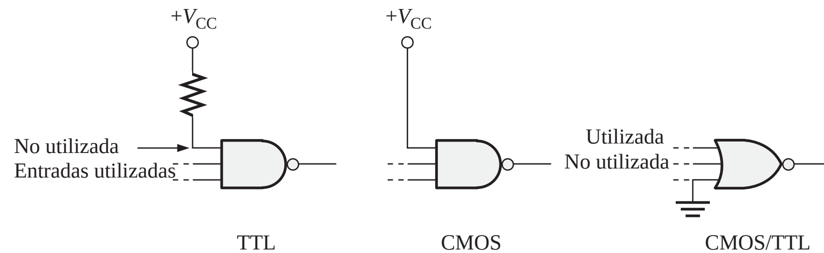
FIGURA 3.66 Parte de una hoja de características de un 74HC00A.

Hojas de características

Una hoja de características típica consta de una página de información que muestra, entre otras cosas, el diagrama lógico y los encapsulados, las condiciones de operación recomendadas, las características eléctricas y las características de conmutación. En las Figuras 3.65 y 3.66 se muestran, respectivamente, parte de las hojas de características para un 74LS00 y un 74HC00A. La longitud de las hojas de características varía y unas aportan mucha más información que otras.

CONSEJOS PRÁCTICOS

Las entradas de puerta no utilizadas para TTL y CMOS deberían conectarse al nivel lógico apropiado (ALTO o BAJO). Para las puertas AND y NAND, es recomendable que las entradas no utilizadas se conecten a V_{CC} (a través de una resistencia de 1,0 kΩ para TTL), y para las puertas OR y NOR, las puertas no utilizadas deberían conectarse a tierra.



REVISIÓN DE LA SECCIÓN 3.8

1. Enumere los dos tipos de tecnologías de circuitos integrados más ampliamente utilizadas.
2. Identifique las siguientes designaciones lógicas de circuitos integrados:

   (a) LS **(b)** ALS **(c)** F **(d)** HC **(e)** AC **(f)** HCT **(g)** LV
3. Identifique los siguientes dispositivos de acuerdo a su función lógica:

   (a) 74LS04 **(b)** 74HC00 **(c)** 74LV08 **(d)** 74ALS10

   (e) 7432 **(f)** 74ACT11 **(g)** 74AHC02
4. Generalmente, ¿qué tecnología de circuitos integrados tienen la disipación de potencia más baja?
5. ¿Qué quiere decir el término *inversor séxtuple*? ¿Y el término *cuádruple NAND de 2 entradas*?
6. Se aplica un pulso positivo a una entrada inversora. El tiempo desde el flanco de subida de la entrada hasta el flanco de subida de la salida es 10 ns. El tiempo desde el flanco de bajada de la entrada hasta el flanco de bajada de la salida es de 8 ns. ¿Cuáles son los valores de t_{PLH} y t_{PHL}?
7. Una determinada puerta tiene un retardo de propagación de 6 ns y una disipación de potencia de 3 mW. Determinar el producto velocidad–potencia.
8. Defina I_{CCL} e I_{CCH}.
9. Defina V_{IL} y V_{IH}.
10. Defina V_{OL} y V_{OH}.

3.9 LOCALIZACIÓN DE AVERÍAS

La localización de averías es el proceso de reconocer, aislar y corregir un fallo en un sistema o circuito. Para poder localizar las averías de forma efectiva, debe entender cómo se supone que trabaja el circuito o sistema y debe estar en disposición de reconocer un funcionamiento incorrecto. Por ejemplo, para determinar si una puerta lógica tiene un fallo, debe saber cuál debe ser la salida para unas entradas dadas.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Comprobar la existencia de entradas y salidas abiertas internamente en las puertas lógicas de los CI. ■ Reconocer los efectos de una entrada o una salida del CI cortocircuitada. ■ Detectar en una tarjeta de circuito impreso la existencia de fallos externos. ■ Localizar las averías en un sencillo contador de frecuencia utilizando un osciloscopio.

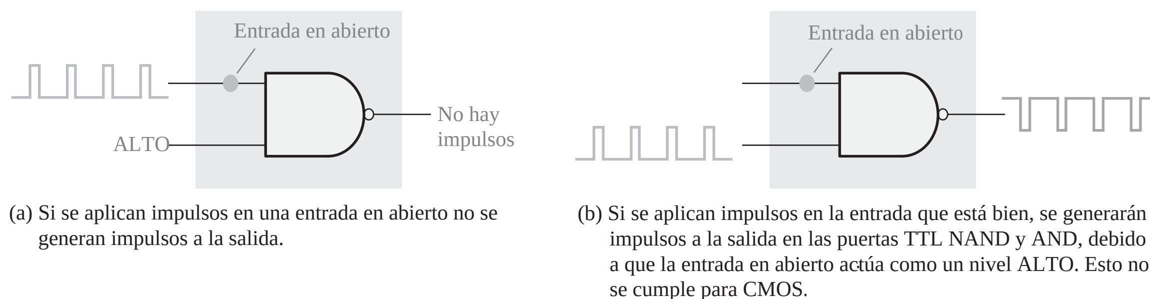
Fallos internos en las puertas lógicas de los CI

Los circuitos abiertos y los cortocircuitos son los fallos más comunes en las puertas internas del CI. Se pueden producir tanto en las entradas como en la salida de una puerta contenida en el encapsulado del CI. *Antes de intentar solucionar cualquier avería, compruebe que la alimentación continua y la masa son correctas.*

Efectos de una entrada que se encuentra en circuito abierto internamente. Un circuito abierto interno es el resultado de un componente en circuito abierto o de una ruptura en la conexión entre el chip y el pin del encapsulado. Una entrada en circuito abierto impide que una señal de impulsos en esta entrada dé lugar a una salida, como se muestra en la Figura 3.67(a) para la puerta NAND de 2 entradas. Una entrada TTL en abierto actúa como un nivel ALTO, por lo que los impulsos aplicados a la entrada que está en buen estado pasan a través de la puerta NAND hasta la salida, como se muestra en la Figura 3.67(b).

Condiciones para probar las puertas. Al probar una puerta NAND o una puerta AND, debe asegurarse siempre de que las entradas a las que no se aplican impulsos se encuentren a nivel ALTO, para activar la puerta. Cuando pruebe una puerta NOR o una puerta OR, debe asegurarse siempre de que las entradas a las que no se aplican impulsos se encuentran a nivel BAJO. Cuando se prueba una puerta XOR o XNOR, el nivel de la entrada a la que no se aplican impulsos no importa, ya que los impulsos aplicados en la otra entrada forzarán a que las entradas se encuentren, alternativamente, en el mismo nivel o en niveles opuestos.

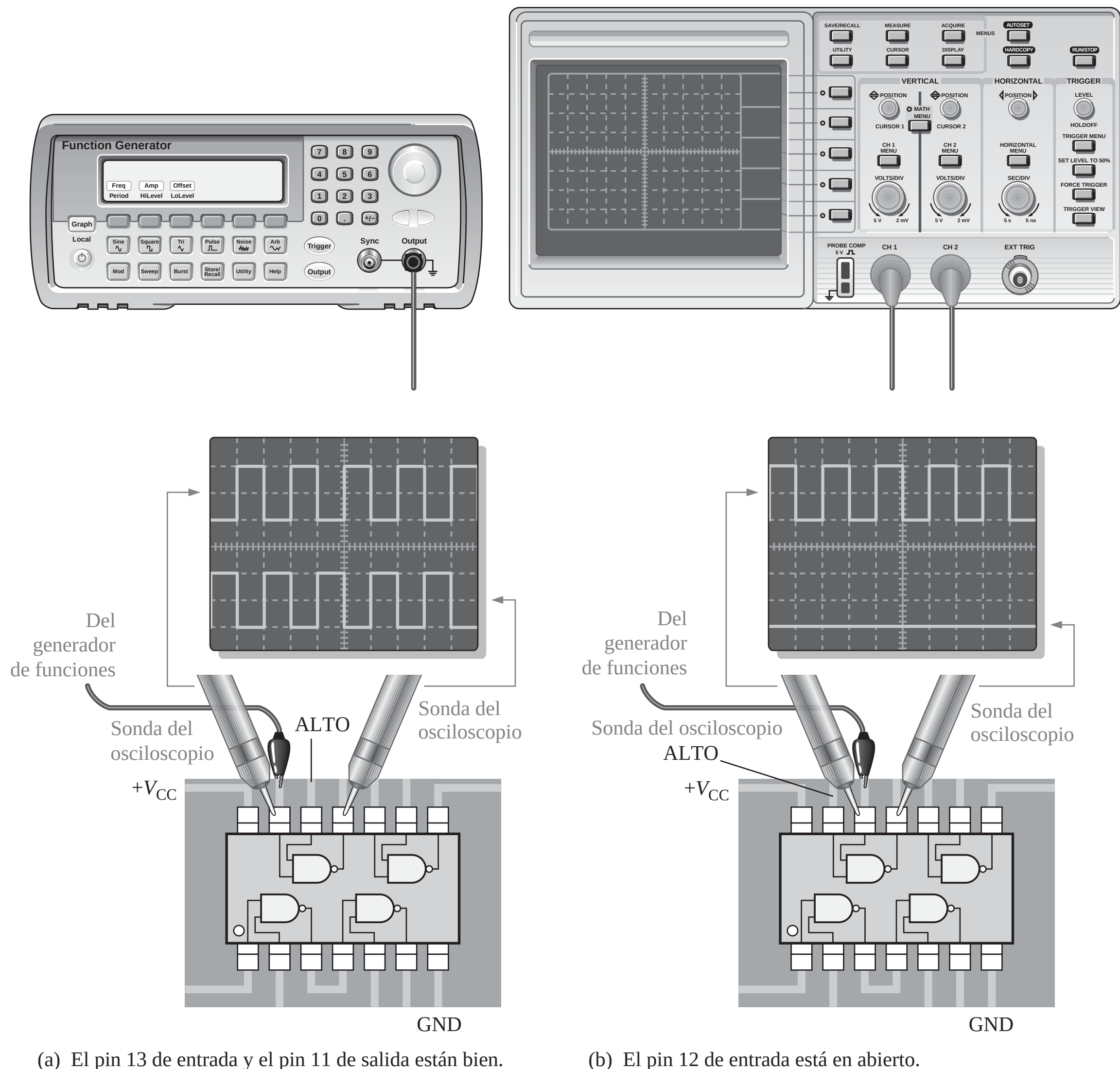
Localización de fallo: entrada en circuito abierto. La localización de este tipo de fallo es, en la mayoría de los casos, muy fácil utilizando un osciloscopio y un generador de funciones, como se muestra en la Figura 3.68, para el caso de una puerta NAND de 2 entradas. Al medir las señales digitales con un osciloscopio, emplee siempre el acoplamiento en continua.



(a) Si se aplican impulsos en una entrada en abierto no se generan impulsos a la salida.

(b) Si se aplican impulsos en la entrada que está bien, se generarán impulsos a la salida en las puertas TTL NAND y AND, debido a que la entrada en abierto actúa como un nivel ALTO. Esto no se cumple para CMOS.

FIGURA 3.67 Efecto de una entrada en circuito abierto en una puerta NAND.



(a) El pin 13 de entrada y el pin 11 de salida están bien.

(b) El pin 12 de entrada está en abierto.

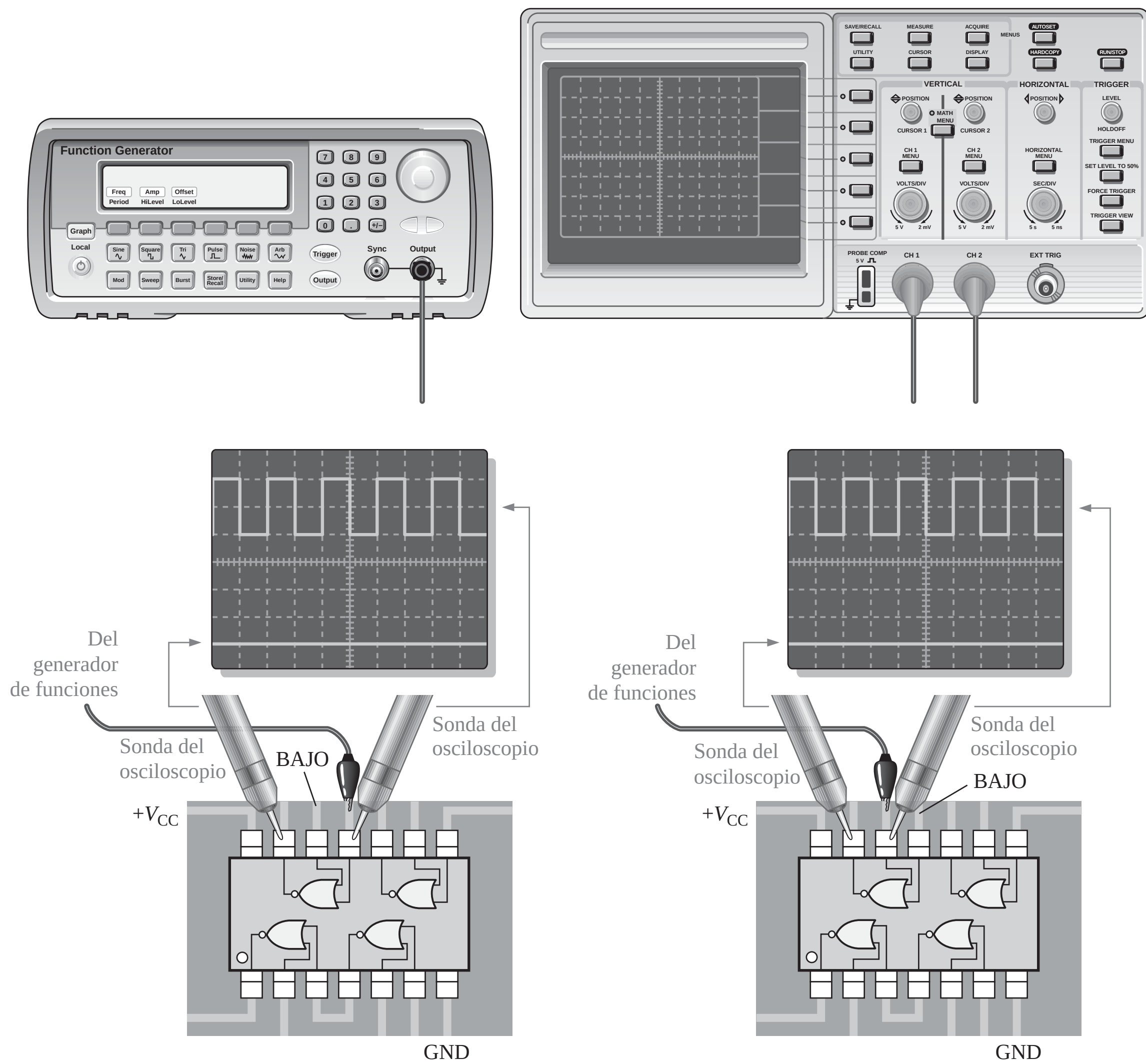
FIGURA 3.68 Localización de averías en una puerta NAND con una entrada en circuito abierto.

El primer paso en la localización de averías de un CI, cuando se sospecha que está fallando, es asegurarse de que la tensión de alimentación continua (V_{CC}) y la masa están conectadas a los pines apropiados del CI. Después, se aplican impulsos continuos a una de las entradas de la puerta, asegurándose de que las otras entradas están a nivel ALTO (en el caso de una puerta NAND). En la Figura 3.68(a), se comienza a aplicar los impulsos en el pin 13, ya que se ha determinado que es una de las entradas de la puerta de la que se sospecha el fallo. Si en la salida correspondiente (en este caso el pin 11) se detecta un tren de impulsos, entonces el pin 13 de entrada no está en abierto. Consecuentemente, esto prueba también que la salida no está en abierto. A continuación, se aplica otro tren de impulsos a otra entrada de la puerta (pin 12), asegurándose de que la otra entrada está a nivel ALTO. En la salida (en el pin 11) no se detecta un tren de impulsos y la salida está a nivel BAJO, lo que indica que la entrada del pin 12 está en abierto, como se muestra en la Figura 3.68(b). Observe que la entrada en la que no se aplican impulsos debe estar a nivel ALTO en el caso de una puerta NAND o

una puerta AND. Si se tratara de una puerta NOR, la entrada en la que no se aplican impulsos debería estar a nivel BAJO.

Efectos de una salida que se encuentra internamente en circuito abierto. Una salida que esté internamente en circuito abierto impide que una señal en cualquiera de las entradas llegue hasta la salida. Por tanto, independientemente de las condiciones de entrada, la salida no se ve afectada. El nivel en el pin de salida del CI dependerá de a qué esté conectada la salida externamente, por lo que podría estar a nivel ALTO, BAJO o flotante (no fijado a ninguna referencia). En cualquier caso, no habrá señal en el pin de salida.

Localización de fallo: salida en abierto. La Figura 3.69 ilustra la localización de una salida en abierto en una puerta NOR. En la parte (a) de la figura, se aplican impulsos a una de las entradas de las que se sospecha (en



(a) Se aplica un impulso de entrada en el pin 11. No hay impulsos en la salida

(b) Se aplica un impulso de entrada en el pin 12. No hay impulsos en la salida

FIGURA 3.69 Localización de una salida en circuito abierto en una puerta NOR.

este caso, el pin 11), y la salida (pin 13) no presenta ningún impulso. En la parte (b) de la figura, se aplican impulsos a la otra entrada (pin 12) y, de nuevo, no hay indicación de impulsos en la salida. Bajo la condición de que en la entrada en la que no se aplican impulsos esté a nivel BAJO, esta prueba demuestra que la salida está internamente en circuito abierto.

Entrada o salida cortocircuitadas. Aunque no es un fallo común como un abierto, se puede producir un cortocircuito interno a la tensión de alimentación, a masa, a otra entrada o a una salida. Cuando una entrada o una salida se cortocircuita a la alimentación, permanecerá en estado ALTO. Si una entrada o una salida se cortocircuita a masa, permanecerá a nivel BAJO (0 V). Si dos entradas o una entrada y una salida se cortocircuitan entre sí, entonces estarán siempre al mismo nivel.

Abiertos y cortos externos

Muchos de los fallos que afectan a los CI digitales se deben a fallos externos a los mismos. Se incluyen en este tipo de fallos las soldaduras incorrectas, salpicaduras de estaño, cortes de conductores, tarjetas de circuito

EJEMPLO 3.24

Se quiere probar un 74LS10, una triple puerta NAND de 3 entradas, que es uno de los muchos CI que contiene una tarjeta de circuito impreso. Ha probado los pines 1 y 2, y ambos están a nivel ALTO. Después, aplica un tren de impulsos al pin 13, y coloca la sonda del osciloscopio, en primer lugar, en el pin 12 y luego en la pista del circuito impreso, como se indica en la Figura 3.70. Basándose en la observación de la pantalla del osciloscopio, ¿cuál es, probablemente, el principal problema?

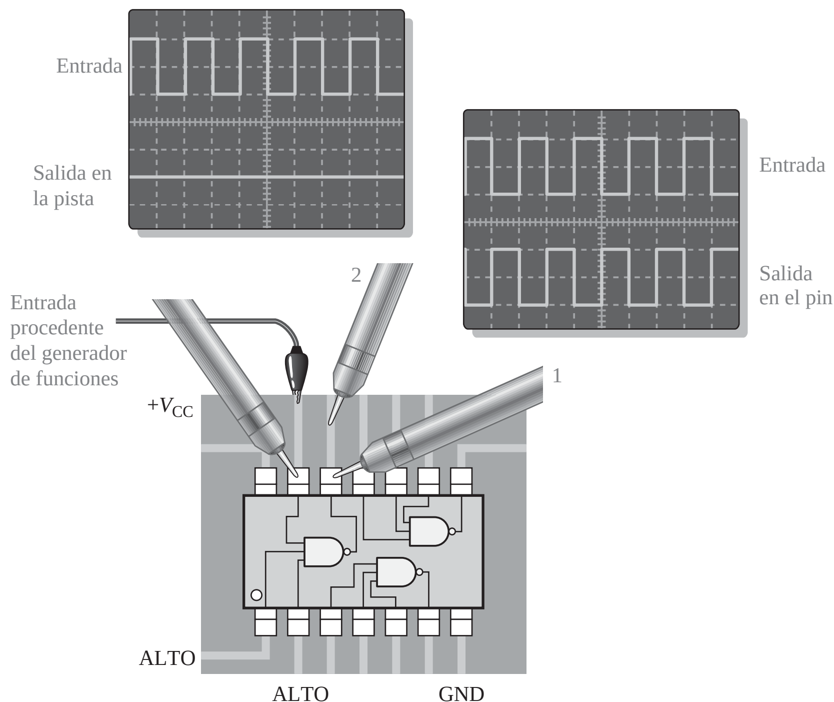


FIGURA 3.70

Solución La forma de onda cuando se coloca la sonda en la posición 1 muestra que hay actividad de impulsos en la salida de la puerta en el pin 12 pero, sin embargo, no detecta actividad en la pista de la tarjeta de circuito impreso, como indica la sonda en la posición 2. La puerta está trabajando correctamente, pero la señal no pasa del pin 12 del CI a la pista del circuito impreso.

Lo más probable es que haya una mala soldadura entre el pin 12 del CI y la tarjeta de circuito impreso, lo que da lugar a un circuito abierto. Por tanto, debería volver a soldar este punto y probar de nuevo.

Problema relacionado Si no se detectan impulsos en ningún punto de la Figura 3.70, ¿qué fallo(s) indica esto?

impreso grabadas incorrectamente, y rupturas o interrupciones en las conexiones o en las interconexiones del circuito impreso. Estas condiciones de circuitos abiertos o cortocircuitos tienen los mismos efectos sobre las puertas lógicas que los fallos internos y, básicamente, se localizan mediante los mismos métodos. Cuando se sospecha que algo está fallando, lo primero que debe hacer un técnico es una inspección visual del circuito.

Casi siempre deberá localizar fallos en circuitos integrados que están montados en tarjetas de circuito impreso o en prototipos ensamblados, y están interconectados con otros circuitos integrados. Según vaya avanzando a través de este libro, aprenderá cómo se usan diferentes tipos de CI digitales conjuntamente para realizar funciones de sistemas. Sin embargo, en este momento, vamos a concentrarnos en las puertas individuales de los circuitos digitales. Esta limitación no nos impide abordar el concepto de sistema en un nivel básico y simplificado.

Para continuar con el concepto de sistemas, los Ejemplos 3.25 y 3.26 se ocupan de la localización de averías en el contador de frecuencia introducido en la Sección 3.2.

EJEMPLO 3.25

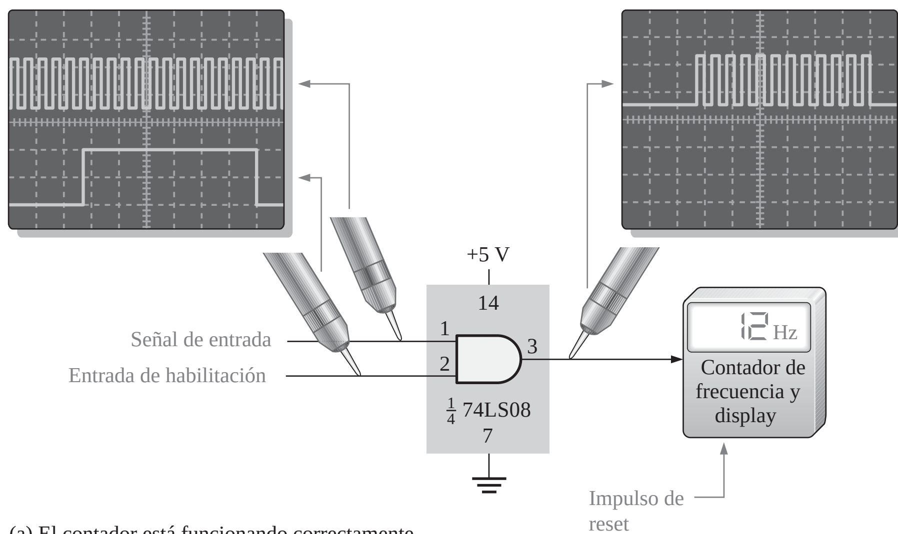
Después de intentar trabajar con el contador de frecuencia de la Figura 3.71, se encuentra con que constantemente da una lectura de todo 0s en el display, independientemente de la frecuencia de entrada. Determinar la causa de este mal funcionamiento. El impulso de habilitación tiene una anchura de 1 s.

La Figura 3.71(a) proporciona un ejemplo de cómo debería funcionar el contador de frecuencia con un tren de impulsos a 12 Hz aplicado en la entrada de la puerta AND. La parte (b) de la figura muestra que el display indica, incorrectamente, 0 Hz.

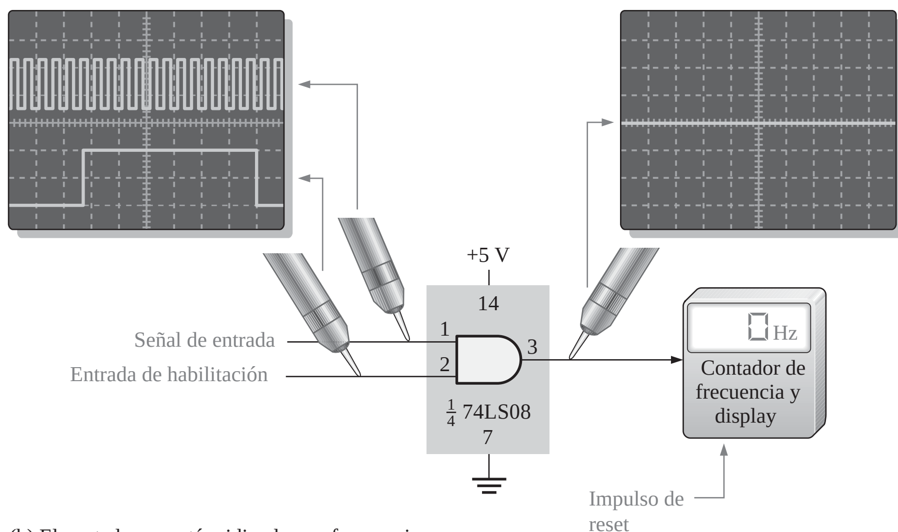
Solución Existen tres posibles causas:

1. Un nivel activo o verdadero constante en la entrada reset del contador hace que el contador esté siempre a cero.
2. No hay tren de impulsos en la entrada del contador, debido a que existe un circuito abierto o un cortocircuito en el contador. Este problema impediría que el contador avanzará después de haber sido puesto a cero.
3. No hay tren de impulsos a la entrada del contador porque hay un circuito abierto en la salida de la puerta AND o por la ausencia de señales de entrada, lo que de nuevo impide que el contador avance a partir de cero.

El primer paso consiste en asegurarse de que V_{CC} y masa están correctamente conectadas en todos los puntos; supongamos que están bien. A continuación, probamos aplicando impulsos en ambas entradas de la puerta AND. El osciloscopio indica que hay impulsos en ambas entradas. Se com-



(a) El contador está funcionando correctamente.



(b) El contador no está midiendo una frecuencia.

FIGURA 3.71

prueba la entrada reset del contador y muestra un nivel BAJO, que se sabe que corresponde a un nivel no verdadero y, por tanto, aquí no está el problema. La siguiente prueba se hace en el pin 3 del 74LS08, y muestra que no hay impulsos en la salida de la puerta AND, lo que indica que la salida de la puerta está internamente en circuito abierto. Debe reemplazar el CI 74LS08 y probar de nuevo el funcionamiento.

Problema relacionado Si el pin 2 de la puerta AND 74LS08 estuviese en circuito abierto, ¿qué indicación vería en el display?

EJEMPLO 3.26

El contador de frecuencia mostrado en la Figura 3.72 parece medir incorrectamente la frecuencia de las señales de entrada. Se determina que, cuando se aplica al pin 1 de la puerta AND una señal con una frecuencia conocida, la pantalla del osciloscopio indica una frecuencia más alta. Determinar dónde está el fallo. Las lecturas de la pantalla están expresadas en segundos/división.

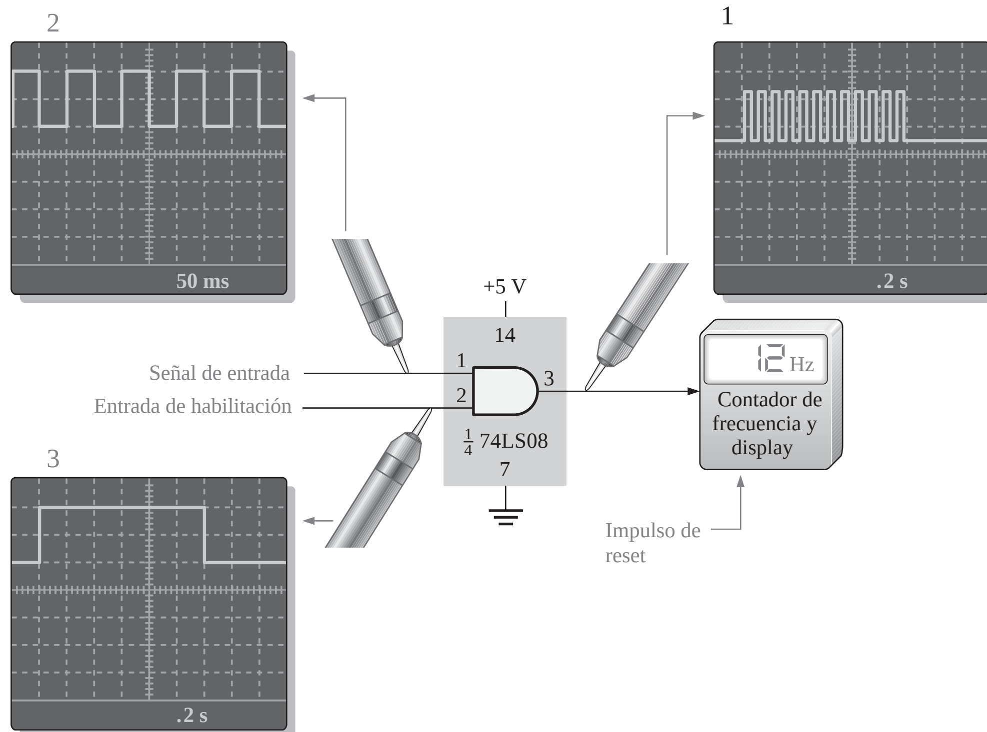


FIGURA 3.72

Solución Recuerde de la Sección 3.2 que los impulsos de entrada pueden pasar a través de la puerta AND exactamente durante 1 s. El número de impulsos que se cuentan en 1 s es igual a la frecuencia en hertzios (ciclos por segundo). Por tanto, el intervalo de 1 s, que es generado por el impulso de habilitación aplicado en el pin 2 de la puerta AND, es crítico para obtener una medida de frecuencia precisa. Los impulsos de habilitación se generan internamente mediante un circuito oscilador de precisión. El ancho del impulso debe ser exactamente de 1 s y, en este caso, se produce cada 3 s para actualizar el contador. Justo después de cada impulso de habilitación, el contador se pone a cero, por lo que empieza a contar de nuevo.

Puesto que el contador cuenta más impulsos que los que debería, dando lugar a que se obtenga una lectura de frecuencia más alta, parece que el impulso de habilitación es el sospechoso principal. La medida exacta del intervalo de tiempo debe hacerse con el osciloscopio.

	Se aplica al pin 1 de la puerta AND un tren de impulsos de entrada de exactamente 10 Hz e, incorrectamente, el display presenta 12 Hz. La primera medida en el osciloscopio, en la salida de la puerta AND, muestra que hay 12 impulsos por cada impulso de activación. La segunda medida del osciloscopio verifica que la frecuencia de entrada es exactamente 10 Hz (período = 100 ms). La tercera medida del osciloscopio determina el ancho del impulso de habilitación, que es 1,2 s, mayor por tanto que 1 s. La conclusión es que el impulso de habilitación no está bien calibrado por alguna razón.
Problema relacionado	¿De qué sospecharía si la lectura indicara una frecuencia menor de lo que debe ser?

CONSEJOS PRÁCTICOS

La correcta puesta a tierra es muy importante cuando se prepara un circuito para tomar medidas o trabajar en él. La correcta puesta a tierra del osciloscopio protege al usuario frente a descargas eléctricas y la conexión a tierra del propio usuario protege los circuitos frente a posibles daños. Poner a tierra el osciloscopio quiere decir conectarlo a la toma de masa, introduciendo el conector de tres tomas en un enchufe con toma de masa. Como ya sabrá, ponerse a tierra el usuario significa que debe utilizarse una pulsera de conexión a masa, especialmente cuando se trabaja con circuitos CMOS.

Para obtener medidas precisas, también debe asegurarse de que la tierra del circuito que se está probando sea la misma que la del osciloscopio. Esto puede hacerse conectando el terminal de tierra de la sonda del osciloscopio a un punto de tierra conocido del circuito, como puede ser el chasis metálico o un punto de tierra en la tarjeta de circuito impreso. También se puede conectar la tierra del circuito al conector GND ubicado en el panel frontal del osciloscopio.

REVISIÓN DE LA SECCIÓN 3.9

1. ¿Cuáles son los tipos de fallos más comunes en los circuitos integrados?
2. Si se aplican dos señales de entrada diferentes a una puerta NAND TTL de 2 entradas y la señal de salida es como una de las entradas, pero invertida, ¿cuál es el problema más probable?
3. Cite dos características de los trenes de impulsos que puedan medirse con un osciloscopio.

RESUMEN

- La salida de un inversor es el complemento de la entrada.
- La salida de una puerta AND es un nivel ALTO sólo si todas las entradas están a nivel ALTO.
- La salida de una puerta OR es un nivel ALTO si cualquiera de las entradas está a nivel ALTO.
- La salida de una puerta NAND es un nivel BAJO sólo si todas las entradas están a nivel ALTO.
- La puerta NAND puede expresarse como una puerta negativa–OR, cuya salida es un nivel ALTO cuando cualquier entrada está a nivel BAJO.
- La salida de una puerta NOR es un nivel BAJO si cualquiera de las entradas está a nivel ALTO.
- La puerta NOR puede expresarse como una puerta negativa–AND, cuya salida es un nivel ALTO sólo si todas las entradas están a nivel BAJO.
- La salida de una puerta OR–exclusiva es un nivel ALTO cuando las entradas son distintas.
- La salida de una puerta NOR–exclusiva es un nivel BAJO cuando las entradas son distintas.

- Los símbolos distintivos y tablas de verdad para los distintos tipos de puertas lógicas (sólo para dos entradas) se muestran en la Figura 3.73.

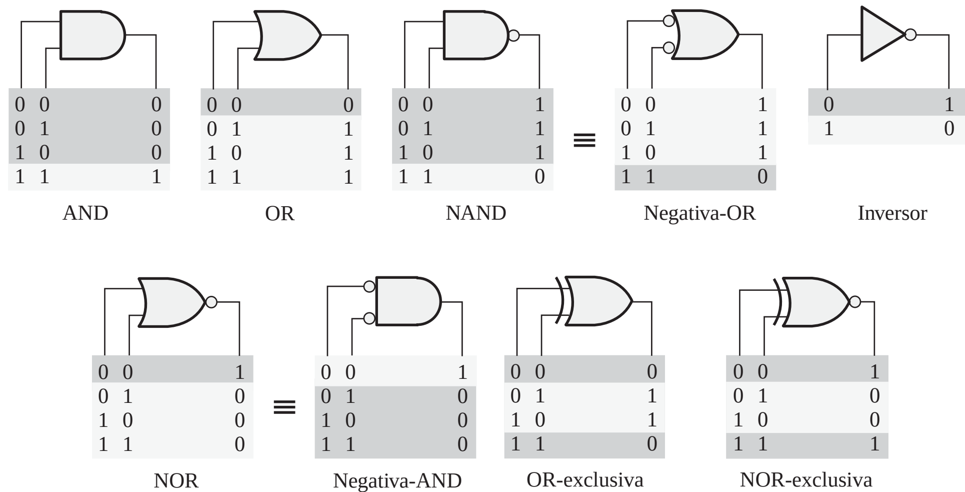


FIGURA 3.73

- La mayoría de los dispositivos lógicos programables (PLD) están basados en alguna forma de matriz AND.
- Las tecnologías basadas en conexiones programables son las tecnologías basadas en fusibles, antifusibles, EPROM, EEPROM y SRAM.
- Un PLD puede programarse utilizando una herramienta hardware denominada programador o montado en una tarjeta de circuito impreso de desarrollo.
- Los PLD tienen asociado un paquete de desarrollo software para la tarea de programación.
- Los dos métodos disponibles para introducir el diseño en un software de programación son la interfaz de texto (HDL) y la interfaz gráfica (esquemáticos)
- Los PLD de programación dentro del sistema (ISP) pueden programarse después de ser instalados en un sistema.
- JTAG (*Joint Test Action Group*) es una interfaz estándar (IEEE Std. 1149.1) utilizada para programar y probar los dispositivos PLD.
- La tecnología CMOS se basa en los transistores de efecto de campo MOS.
- La tecnología TTL se basa en transistores de unión bipolares.
- Por regla general, los dispositivos CMOS tienen el consumo de potencia más bajo que los TTL.
- La disipación media de potencia de una puerta lógica es:

$$P_D = V_{CC}\left(\frac{I_{\text{CCH}} + I_{\text{CCL}}}{2}\right)$$

- El producto velocidad–potencia de una puerta lógica es

$$SPP = t_p P_D$$

PALABRAS CLAVE

Las palabras clave y los términos que se han resaltado en negrita se encuentran en el glosario final del libro.

Álgebra booleana Las matemáticas de los circuitos lógicos.

Antifusible Tipo de conexión programable no volátil de un PLD que se puede dejar abierta o se puede cortocircuitar una sola vez según indique el programa.
Carga unidad Una medida del fan–out. Una entrada de puerta representa una carga unidad a la salida de la puerta, dentro de la misma familia de circuitos integrados.
CMOS *Complementary Metal–Oxide Semiconductor*, semiconductor complementario de metal–óxido; un tipo de circuito lógico integrado que se implementa con transistores de efecto de campo.
Complemento El inverso u opuesto de un número. Un nivel BAJO es el complemento de un nivel ALTO y 0 es el complemento de 1.
Diagrama de tiempos Diagrama de señales que muestra las relaciones temporales de todas las señales.
Dispositivo objetivo PLD montado en una herramienta de programación o tarjeta de desarrollo en el que se descarga un diseño lógico hardware.
EPROM Tipo de conexión programable no volátil de un PLD basado en celdas de memoria de sólo lectura eléctricamente programables y que se pueden activar y desactivar repetidamente por programación.
EEPROM Tipo de conexión programable no volátil de un PLD basado en celdas de memoria de sólo lectura eléctricamente programables y borrables, que se pueden activar y desactivar repetidamente por programación.
Fan–out Número de entradas de puertas equivalentes de la misma familia que puede excitar una puerta lógica.
Fusible Tipo de conexión programable no volátil de un PLD que se puede dejar cortocircuitada o se puede dejar en abierto una sola vez según indique el programa
Habilitar Activar o poner en modo operacional. Una entrada de un circuito lógico que activa su funcionamiento.
Inversor Circuito lógico que invierta o complementa su entrada.
Matriz AND Matriz de puertas AND que consta de una matriz de interconexiones programables.
Puerta AND Puerta lógica que produce una salida a nivel ALTO sólo cuando todas las entradas están a nivel ALTO.
Puerta NAND Puerta lógica que produce una salida a nivel BAJO sólo si todas las entradas están a nivel ALTO.
Puerta NOR Puerta lógica en la que la salida es un nivel BAJO cuando al menos una de las entradas está a nivel ALTO.
Puerta NOR–exclusiva (XNOR) Puerta lógica que produce una salida a nivel BAJO sólo cuando las dos entradas tienen niveles opuestos.
Puerta OR Puerta lógica que produce una salida a nivel ALTO cuando una o más entradas están a nivel ALTO.
Puerta OR–exclusiva (XOR) Puerta lógica que produce una salida a nivel ALTO sólo cuando las dos entradas tienen niveles opuestos.
SRAM Tipo de conexión programable volátil de PLD basada en celdas de memoria de acceso aleatorio y que se pueden activar u desactivar repetidamente mediante programación.
Tiempo de retardo de propagación Intervalo de tiempo que transcurre entre la transición de entrada y su correspondiente transición de salida en un circuito lógico.
Tabla de verdad Tabla que muestra las entradas y los correspondientes niveles de salida de un circuito lógico.
TTL *Transistor–Transistor Logic*, lógica transistor–transistor, un tipo de circuito integrado que utiliza transistores de unión bipolares.

AUTOTEST

Las respuestas se encuentran al final del capítulo

1. Cuando la entrada de un inversor es un nivel ALTO (1), la salida es:

(a) un nivel ALTO o 1 **(b)** un nivel BAJO o 1
(c) un nivel ALTO o 0 **(d)** un nivel BAJO o 0

2. Un inversor realiza la operación conocida como:
(a) complementación **(b)** afirmación
(c) inversión **(d)** las respuestas (a) y (c)

3. La salida de una puerta AND con entradas *A*, *B*, y *C* está a 1 (nivel ALTO) cuando
(a) A = 1, B = 1, C =1 **(b)** A = 1, B = 0, C =1 **(c)** A = 0, B = 0, C =0

4. La salida de una puerta OR con entradas *A*, *B*, y *C* está a 1 (nivel ALTO) cuando
(a) $A = 1, B = 1, C = 1$ **(b)** $A = 0, B = 0, C = 1$ **(c)** $A = 0, B = 0, C = 0$
(d) las respuestas (a), (b) y (c) **(e)** sólo las respuestas (a) y (b)

5. Se aplica un impulso a cada una de las entradas de una puerta NAND de dos entradas. Para $t = 0$ un impulso está a nivel ALTO y pasa a nivel BAJO en el instante $t = 1$ ms. El otro impulso está a nivel ALTO en $t = 0{,}8$ ms y pasa a nivel BAJO en $t = 3$ ms. El impulso de salida puede describirse como sigue:
(a) A nivel BAJO en $t = 0$, y pasará a nivel ALTO en $t = 3$ ms.
(b) A nivel BAJO en $t = 0{,}8$ ms, y pasará a nivel ALTO en $t = 3$ ms.
(c) A nivel BAJO en $t = 0{,}8$ ms, y pasará a nivel ALTO en $t = 1$ ms.
(d) A nivel ALTO en $t = 0{,}8$ ms, y pasará a nivel BAJO en $t = 1$ ms.

6. Se aplica un impulso a cada una de las entradas de una puerta NOR de dos entradas. Para $t = 0$ un impulso está a nivel ALTO y pasa a nivel BAJO en el instante $t = 1$ ms. El otro impulso está a nivel ALTO en $t = 0{,}8$ ms y pasa a nivel BAJO en $t = 3$ ms. El impulso de salida puede describirse como sigue:
(a) A nivel BAJO en $t = 0$, y pasará a nivel ALTO en $t = 3$ ms.
(b) A nivel BAJO en = 0,8 ms, y pasará a nivel ALTO en $t = 3$ ms.
(c) A nivel BAJO en $t = 0{,}8$ ms, y pasará a nivel ALTO en $t = 1$ ms.
(d) A nivel ALTO en $t = 0{,}8$ ms, y pasará a nivel BAJO en $t = 1$ ms.

7. Se aplica un impulso a cada una de las entradas de una puerta OR–exclusiva. Para $t = 0$ un impulso está a nivel ALTO y pasa a nivel BAJO en el instante $t = 1$ ms. El otro impulso está a nivel ALTO en $t = 0{,}8$ ms y pasa a nivel BAJO en $t = 3$ ms. El impulso de salida estará:
(a) A nivel BAJO en $t = 0$, y pasará a nivel BAJO en $t = 3$ ms.
(b) A nivel BAJO en $t = 0$ ms, y pasará a nivel BAJO en $t = 0{,}8$ ms.
(c) A nivel ALTO en $t = 1$ ms, y pasará a nivel ALTO en $t = 3$ ms.
(d) las respuestas (b) y (c)

8. Se aplica un impulso positivo a un inversor. El intervalo de tiempo entre el flanco anterior de la entrada y el flanco anterior de la salida es 7 ns. Este parámetro es
(a) el producto velocidad–potencia **(b)** el retardo de propagación t_{PHL}.
(c) el retardo de propagación t_{PLH}. **(d)** el ancho del impulso

9. El propósito de una conexión programable en una matriz AND es
(a) conectar una variable de entrada a una entrada de puerta
(b) conectar una fila a una columna de la matriz de interconexiones
(c) desconectar una fila de una columna de la matriz de interconexiones
(d) hacer todo lo anterior

10. El término OTP significa
(a) *Open Test Point* **(b)** *One–Time Programmable*

(c) *Output Test Program* **(d)** *Output Terminal Positive*

11. Los tipos de tecnologías de proceso basadas en conexiones programables son:
(a) antifusible **(b)** EEPROM **(c)** ROM
(d) las respuestas (a) y (b) **(e)** las respuestas (a) y (c)

12. Una tecnología de proceso volátil basada en conexiones programables es:
(a) fusible **(b)** EPROM
(c) SRAM **(d)** EEPROM

13. Dos métodos de introducir un diseño lógico utilizando un software de desarrollo de PLD son:
(a) interfaz de texto e interfaz numérica **(b)** interfaz de texto e interfaz gráfica
(c) interfaz gráfica y codificación **(d)** compilación y ordenación

14. JTAG es el acrónimo de
(a) *Joint Test Action Group* **(b)** *Java Top Array Group*
(c) *Joint Test Array Group* **(d)** *Joint Time Analysis Group*

15. En la programación dentro del sistema de un PLD normalmente se emplea
(a) un generador de señal de reloj integrado **(b)** un procesador integrado
(c) una PROM integrada **(d)** las respuestas (a) y (b)
(e) las respuestas (b) y (c)

16. Para medir el período de un tren de impulsos, se debe usar
(a) un multímetro digital **(b)** una sonda lógica
(c) un osciloscopio **(d)** un pulsador lógico

17. Una vez medido el período de un tren de impulsos, la frecuencia se calcula
(a) utilizando otro ajuste **(b)** midiendo el ciclo de trabajo
(c) hallando el recíproco del período **(d)** usando otro tipo de instrumento

PROBLEMAS

Las respuestas a los problemas impares se encuentran al final del libro.

SECCIÓN 3.1 El inversor

1. La señal de entrada mostrada en la Figura 3.74 se aplica a un inversor. Dibujar el diagrama de tiempos de la señal de salida respecto a su entrada.



FIGURA 3.74

2. En la Figura 3.75 se muestra una red de inversores en cascada. Si se aplica un nivel ALTO en el punto *A*, determinar los niveles lógicos de los puntos *B* hasta *F*.

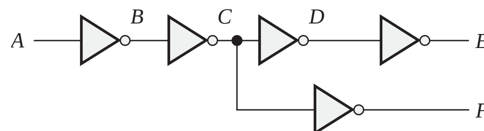


FIGURA 3.75

SECCIÓN 3.2 La puerta AND

3. Determinar la salida *X* para una puerta AND de dos entradas a la que se la aplican las señales de entrada mostradas en la Figura 3.76. Mostrar las relaciones de tiempo de la salida y las entradas mediante un cronograma.

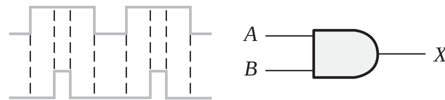


FIGURA 3.76

4. Repetir el problema 3 para las señales de la Figura 3.77.

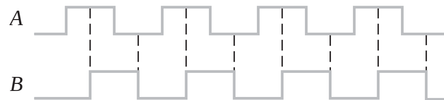


FIGURA 3.77

5. Las señales de entrada que se aplican a una puerta AND de tres entradas son las que se indican en la Figura 3.78. Determinar la señal de salida para las entradas dadas en función del tiempo, utilizando un diagrama de tiempos.

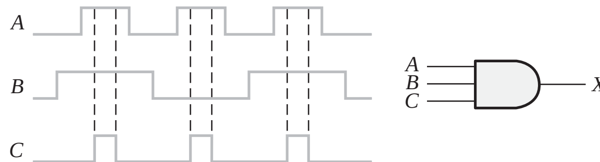


FIGURA 3.78

6. En la Figura 3.79 se indican las señales de entrada que se aplican a una puerta AND de cuatro entradas. Determinar la señal de salida para las entradas dadas en función del tiempo, mediante un cronograma.

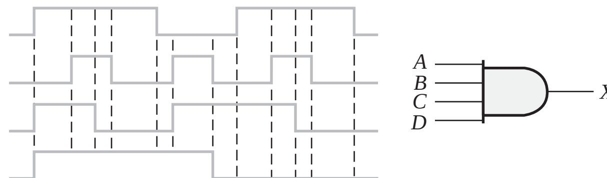


FIGURA 3.79

SECCIÓN 3.3 La puerta OR

7. Determinar la salida de una puerta OR de dos entradas cuando se aplican las señales de entrada dadas en la Figura 3.77 y dibujar el diagrama de tiempos.
8. Repetir el problema 5 para una puerta OR de 3 entradas.
9. Repetir el problema 6 para una puerta OR de 4 entradas.
10. Para las cinco señales de entrada de la Figura 3.80, determinar la salida en una puerta AND de 5 entradas y de una puerta OR de 5 entradas. Dibujar el diagrama de tiempos.

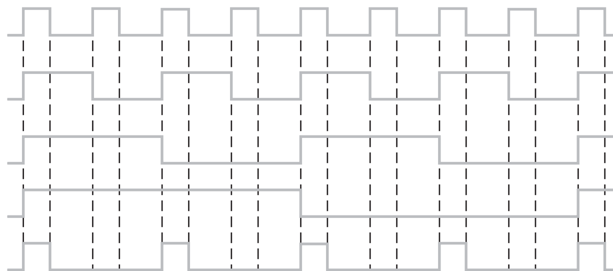


FIGURA 3.80

SECCIÓN 3.4 La puerta NAND

11. Para el conjunto de señales de entrada de la Figura 3.81, determinar la salida de la puerta mostrada y dibujar el diagrama de tiempos.

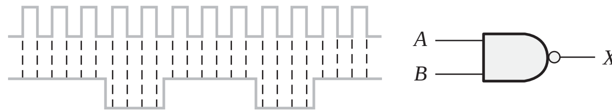


FIGURA 3.81

12. Determinar la salida de la puerta para la señales de entrada de la Figura 3.82 y dibujar el diagrama de tiempos.

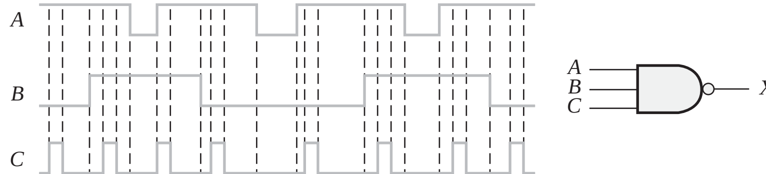


FIGURA 3.82

13. Determinar la señal de la salida correspondiente a la Figura 3.83.

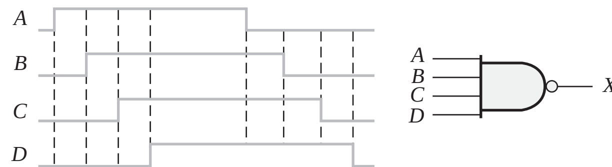


FIGURA 3.83

14. Como ya sabe, los dos símbolos lógicos representados en la Figura 3.84 representan operaciones equivalentes. La diferencia entre ellos es estrictamente de tipo funcional. Para el símbolo NAND, se requieren dos entradas a nivel ALTO para obtener una salida a nivel BAJO. Para el símbolo negativa–OR se requiere al menos una entrada a nivel BAJO para obtener una salida a nivel ALTO. Utilizando estos dos puntos de vista funcionales, demostrar que producirán la misma salida para las entradas dadas.

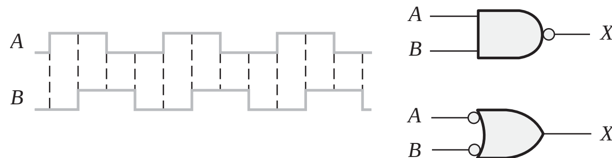


FIGURA 3.84

SECCIÓN 3.5 La puerta NOR

15. Repetir el problema 11 para una puerta NOR de 2 entradas.

16. Determinar la señal de salida para las entradas indicadas en la Figura 3.85, y dibujar el diagrama de tiempos.

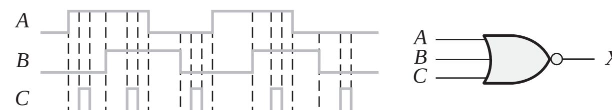


FIGURA 3.85

17. Repetir el problema 13 para una puerta NOR de 4 entradas.

18. Los símbolos de las puertas NAND y negativa–OR representan operaciones equivalentes, pero son funcionalmente diferentes. Para la puerta NOR, se necesita al menos una de las entradas a nivel ALTO para obtener un nivel BAJO de salida. Para la puerta negativa–AND, se necesita que las dos entradas estén a nivel BAJO para obtener un nivel de salida ALTO. Utilizando estos dos puntos de vista funcionales, demostrar que ambas puertas de la Figura 3.86 generarán la misma salida para las entradas dadas.

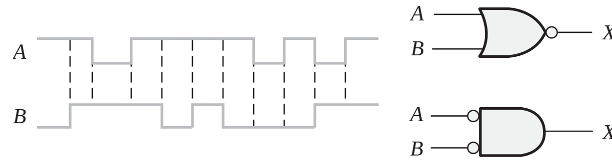


FIGURA 3.86

SECCIÓN 3.6 Puertas OR–exclusiva y NOR–exclusiva

19. ¿En qué difiere la operación lógica de la puerta OR–exclusiva de la puerta OR?

20. Repetir el problema 11 para una puerta OR–exclusiva.

21. Repetir el problema 11 para una puerta NOR–exclusiva.

22. Determinar la salida de una puerta OR–exclusiva para las entradas indicadas en la Figura 3.77, y dibujar el diagrama de tiempos.

SECCIÓN 3.7 Lógica programable

23. En la matriz AND programada mediante conexiones programables de la Figura 3.87, determinar las expresiones booleanas de salida.

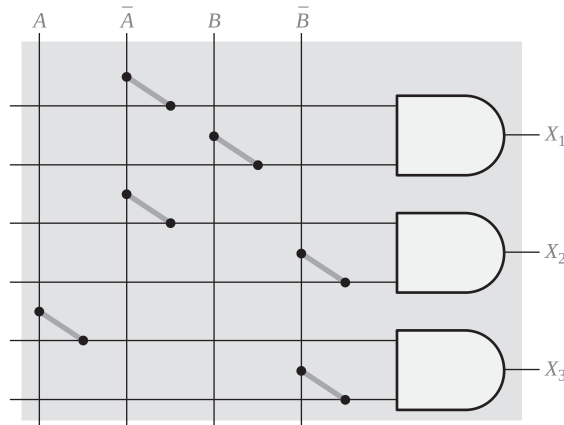


FIGURA 3.87

24. Determinar mediante los números de fila y de columna qué fusibles deben fundirse en la matriz AND programable de la Figura 3.88, para implementar cada uno de los siguientes términos producto: $X_1 = \bar{A}BC$, $X_2 = AB\bar{C}$, $X_3 = \bar{A}B\bar{C}$.

SECCIÓN 3.8 Lógica de función fija

25. En la comparación de ciertos dispositivos lógicos se ha observado que la disipación de potencia para un tipo en concreto aumenta cuando aumenta la frecuencia. ¿Se trata de un dispositivo TTL o CMOS?

26. Utilizando las hojas de características de las Figuras 3.65 y 3.66, determinar lo siguiente:

(a) La disipación de potencia del 74LS00 para la máxima tensión de alimentación y un ciclo de trabajo del 50%.

(b) La tensión de salida mínima para el nivel ALTO de un 74LS00.

(c) El retardo de propagación máximo para un 74LS00.

(d) La tensión de salida máxima para el nivel BAJO de un 74HC00A.

(e) El retardo de propagación máximo para un 74HC00A.

27. Determinar t_{PLH} y t_{PHL} para las señales de la pantalla del osciloscopio de la Figura 3.89. Las lecturas están expresadas en V/div y segundos/división para cada canal.

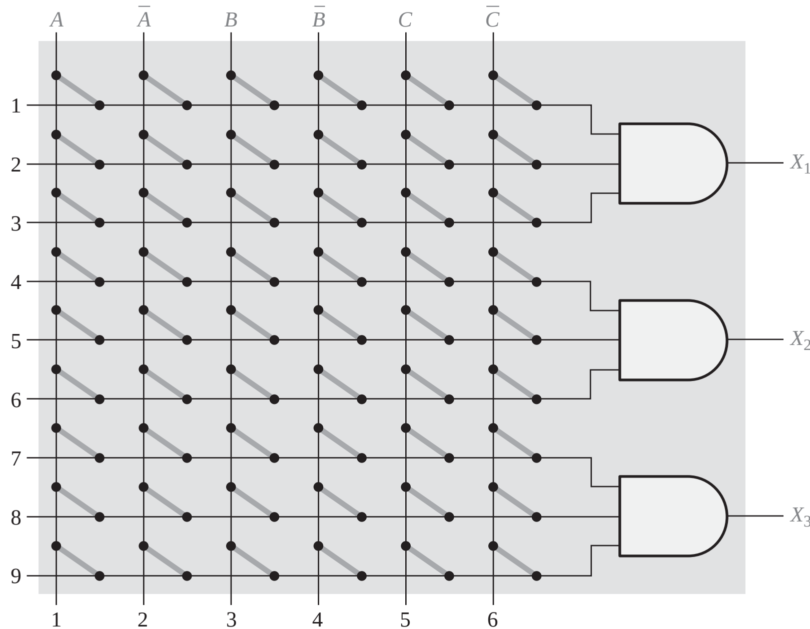


FIGURA 3.88

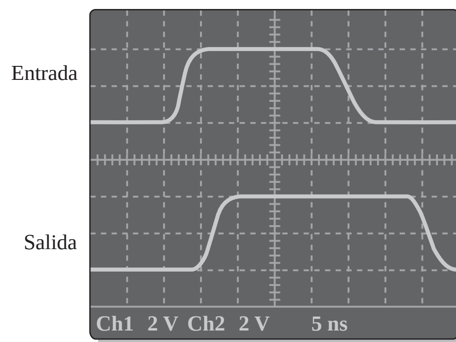


FIGURA 3.89

28. La puerta *A* tiene $t_{PLH} = t_{PHL} = 6$ ns. La puerta *B* tiene $t_{PLH} = t_{PHL} = 10$ ns. ¿Qué puerta puede trabajar a una frecuencia más alta?

29. Si una puerta lógica trabaja con una tensión de alimentación continua de +5 V y circula una corriente media de 4 mA, ¿cuál es la potencia que disipa?

30. La variable I_{CCH} representa la corriente continua de alimentación procedente de V_{CC} cuando todas las salidas de un CI están a nivel ALTO. La variable I_{CCL} representa la corriente de alimentación continua cuando todas las salidas están a nivel BAJO. Para el CI 74LS00, determinar la disipación de potencia típica cuando las salidas de las cuatro puertas están a nivel ALTO. Consulte la hoja de características de la Figura 3.65.

SECCIÓN 3.9 Localización de averías

31. Examinar las condiciones indicadas en la Figura 3.90, e identificar las puertas que fallan.

32. Determinar las puertas que fallan de la Figura 3.91 analizando los cronogramas.

33. Utilizando un osciloscopio, se realizan las observaciones indicadas en la Figura 3.92. Para cada observación, determinar la puerta que es más probable que falle.

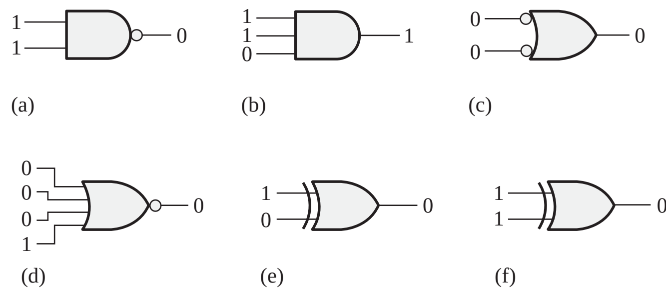


FIGURA 3.90

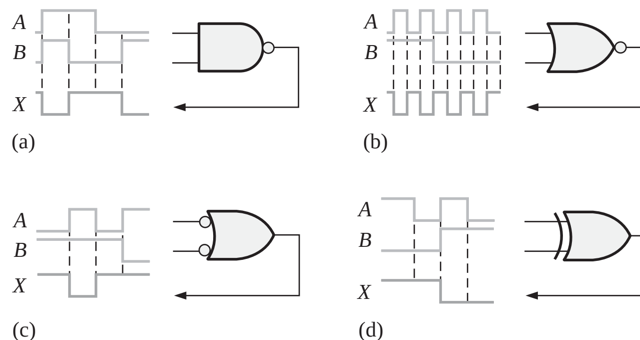


FIGURA 3.91

34. El circuito de alarma de cinturón de seguridad de la Figura 3.16 funciona mal. Se detecta que cuando se enciende el interruptor de arranque y el cinturón está abrochado, la alarma se activa y no se apaga. ¿Cuál será el problema más probable? ¿Cómo lo localizaría?

35. Cada vez que se enciende el interruptor de arranque del circuito de la Figura 3.16, la alarma se activa durante treinta segundos, incluso cuando el cinturón está abrochado. ¿Cuál es la causa más probable de este mal funcionamiento?

36 ¿Qué fallos cree que se pueden haber producido si la salida de una puerta NAND de 3 entradas permanece a nivel ALTO independientemente del nivel de las entradas?



Problemas especiales de diseño

37. Se utilizan sensores para supervisar la presión y la temperatura de una solución química almacenada en un recipiente. La circuitería de cada sensor genera un nivel de tensión ALTO cuando se excede un valor máximo especificado. Cuando se excede la presión o la temperatura, se debe activar una alarma que requiere un nivel de tensión de entrada BAJO. Diseñar un circuito para esta aplicación.

38. En un determinado proceso de fabricación automatizado, se insertan automáticamente los componentes en una tarjeta de circuito impreso. Después de activar la herramienta de inserción, la tarjeta de circuito impreso debe estar correctamente posicionada, y el componente que se va a insertar debe estar en la recámara. Estas condiciones previas se indican mediante un nivel de tensión ALTO. La herramienta de inserción requiere un nivel de tensión BAJO para activarse. Diseñar un circuito para implementar este proceso.

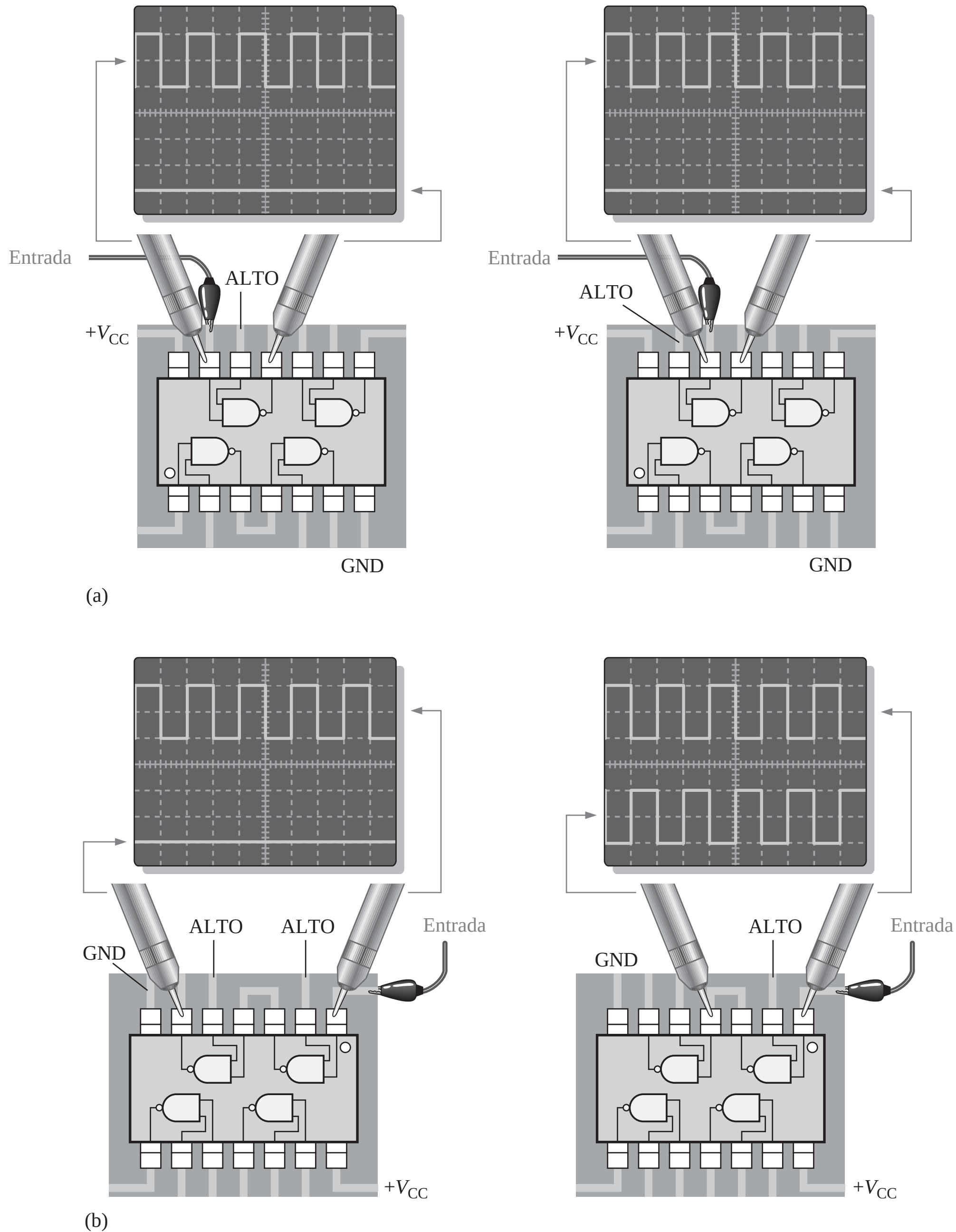


FIGURA 3.92

39. Modificar el contador de frecuencia de la Figura 3.15 para que opere con un impulso de activación (*enable*) que sea activo a nivel BAJO, en lugar de a nivel ALTO, durante el intervalo de 1s.

40. Suponer que la señal de activación de la Figura 3.15 es la forma de onda indicada en la Figura 3.93. Suponer que también se dispone de la señal *B*. Diseñar un circuito que genere un impul-

so de reset activo a nivel ALTO para el contador, sólo durante el tiempo que la señal de activación está a nivel BAJO.

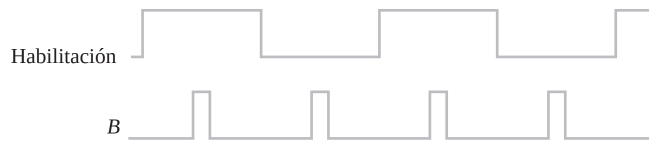


FIGURA 3.93

41. Diseñar un circuito que se colocará en el bloque rayado de la Figura 3.94, que haga que las luces delanteras de un coche se apaguen automáticamente 15 s después de que se apague el interruptor de arranque, en el caso de que el interruptor de las luces se deje activado. Suponer que se necesita un nivel BAJO para apagar las luces.

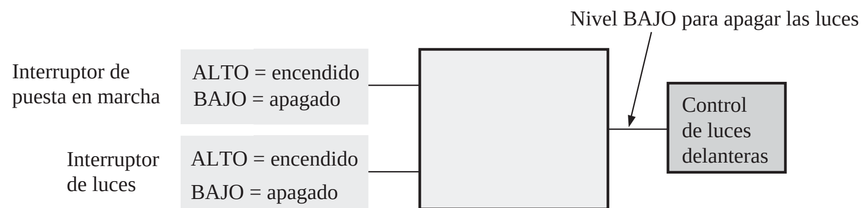


FIGURA 3.94

42. Modificar el circuito lógico de detección de intrusión de la Figura 3.24, para que se puedan proteger dos habitaciones adicionales, cada una de ellas con dos ventanas y una puerta.

43. Modificar el circuito lógico del Problema 42 para realizar un cambio en los sensores de entrada, donde Abierto = nivel BAJO y Cerrado = nivel ALTO.

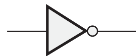
RESPUESTAS

REVISIONES DE CADA SECCIÓN

SECCIÓN 3.1 **El inversor**

1. Cuando la entrada del inversor es 1, la salida es 0.

2. **(a)**



(b) Hay un impulso negativo en la salida (pasa de nivel ALTO a BAJO, y vuelve a nivel ALTO).

SECCIÓN 3.2 **La puerta AND**

1. La salida de una puerta AND es un nivel ALTO cuando todas las entradas están a nivel ALTO.

2. La salida de una puerta AND es un nivel BAJO cuando una o más entradas están a nivel BAJO.

3. Puerta AND de 5 entradas: $X = 1$ cuando $ABCDE = 11111$, y $X = 0$ para las restantes combinaciones de $ABCDE$.

SECCIÓN 3.3 La puerta OR

1. La salida de una puerta OR es un nivel ALTO cuando una o más entradas están a nivel ALTO.
2. La salida de una puerta OR es un nivel BAJO cuando todas las entradas están a nivel BAJO.
3. Puerta OR de 3 entradas: $X = 0$ cuando $ABC = 000$, y $X = 1$ para las restantes combinaciones de ABC.

SECCIÓN 3.4 La puerta NAND

1. La salida de una puerta NAND es un nivel BAJO cuando todas las entradas están a nivel ALTO.
2. La salida de una puerta NAND es un nivel ALTO cuando una o más entradas están a nivel BAJO.
3. NAND: salida activa a nivel BAJO cuando todas las entradas están a nivel ALTO. Negativa–OR: salida activa a nivel ALTO cuando una o más entradas están a nivel BAJO. Ambas tienen la misma tabla de verdad.
4. $X = \overline{ABC}$

SECCIÓN 3.5 La puerta NOR

1. La salida de una puerta NOR es un nivel ALTO cuando todas las entradas están a nivel BAJO.
2. La salida de una puerta NOR es un nivel BAJO cuando una o más entradas están a nivel ALTO.
3. NOR: salida activa a nivel BAJO para una o más entradas a nivel ALTO; negativa–AND: salida activa a nivel ALTO cuando todas las entradas están a nivel BAJO. Ambas tienen la misma tabla de verdad.
4. $X = \overline{A + B + C}$

SECCIÓN 3.6 Puertas exclusiva–OR y exclusiva–NOR

1. La salida de una puerta XOR es un nivel ALTO cuando las entradas están a niveles opuestos.
2. La salida de una puerta XNOR es un nivel ALTO cuando las entradas están al mismo nivel.
3. Aplicar los bits a las entradas de una puerta XOR. Cuando la salida está a nivel ALTO, los bits son diferentes.

SECCIÓN 3.7 Lógica programable

1. Fusible, antifusible, EPROM, EEPROM y SRAM
2. Volátil quiere decir que se pierden todos los datos cuando se desconecta la alimentación y, en consecuencia, el PLD debe reprogramarse; basada en SRAM.
3. Interfaz de texto e interfaz gráfica.
4. JTAG corresponde a *Joint Test Action Group*; el estándar 1149.1 del IEEE para programación y realización de pruebas.

SECCIÓN 3.8 Lógica de función fija

1. CMOS y TTL.
2. **(a)** LS: Schottky de baja potencia. **(b)** ALS: LS avanzada.
   (c) F– TTL rápida **(d)** HC: CMOS de alta velocidad
   (e) AC: CMOS avanzada. **(f)** HCT: HC CMOS TTL compatible
   (g) LV: CMOS de baja tensión.
3. **(a)** 74LS04: inversor séxtuple **(b)** 74HC00: cuádruple NAND de 2 entradas
   (c) 74LV08: cuádruple AND de 2 entradas **(d)** 74ALS10: triple NAND de 3 entradas
   (e) 7432: cuádruple OR de 2 entradas **(f)** 74ACT11: triple AND de 3 entradas

(g) 74AHC02: cuádruple NOR de 2 entradas.

4. Menor disipación de potencia: CMOS.

5. Seis inversores en un encapsulado; cuatro puertas NAND de dos entradas en un encapsulado.

6. t_{PLH}= 10 ns; t_{PHL} = 8 ns.

7. 18 pJ

8. I_{CCL}: corriente de alimentación continua para el estado de salida BAJO; I_{CCH}: corriente de alimentación continua para el estado de salida ALTO.

9. V_{IL}: tensión de entrada para el nivel BAJO; V_{IH}: tensión de entrada para el nivel ALTO.

10. V_{OL}: tensión de salida para el nivel BAJO; V_{OH}: tensión de salida para el nivel ALTO.

SECCIÓN 3.9 Localización de averías

1. Los fallos más comunes son los circuitos abiertos y los cortocircuitos.

2. Una entrada en circuito abierto se comporta como un nivel de entrada ALTO.

3. Amplitud y período.

PROBLEMAS RELACIONADOS

3.1 El diagrama de tiempos no varía.

3.2 *Véase* la Tabla 3.13.

Entradas *ABCD*	Salida *X*	Entradas *ABCD*	Salida *X*
0000	0	1000	0
0001	0	1001	0
0010	0	1010	0
0011	0	1011	0
0100	0	1100	0
0101	0	1101	0
0110	0	1110	0
0111	0	1111	1

TABLA 3.13

3.3 *Véase* la Figura 3.95.

3.4 La forma de onda de salida es igual que la entrada *A*.

3.5 *Véase* la Figura 3.96.

3.6 *Véase* la Figura 3.97.

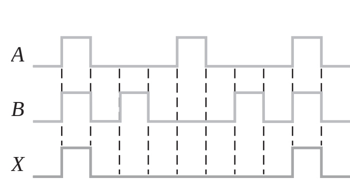


FIGURA 3.95

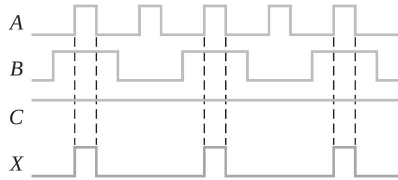


FIGURA 3.96

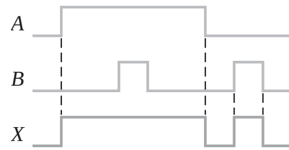


FIGURA 3.97

3.7 *Véase* la Figura 3.98.

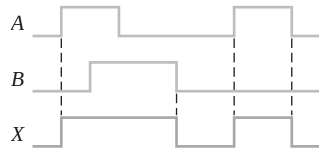


FIGURA 3.98

3.8 *Véase* la Figura 3.99.

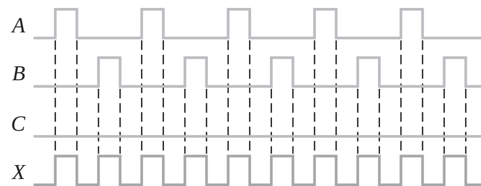


FIGURA 3.99

3.9 *Véase* la Figura 3.100.

3.10 *Véase* la Figura 3.101.

.

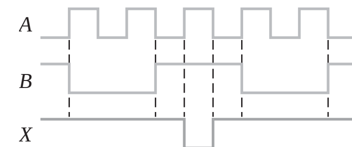


FIGURA 3.100

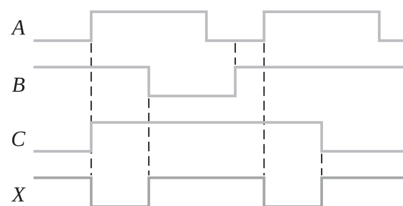


FIGURA 3.101

3.11 Utilizar una puerta NAND de 3 entradas.

3.12 Utilizar una puerta NAND de 4 entradas que funcione como una puerta OR–Negativa.

3.13 *Véase* la Figura 3.102.

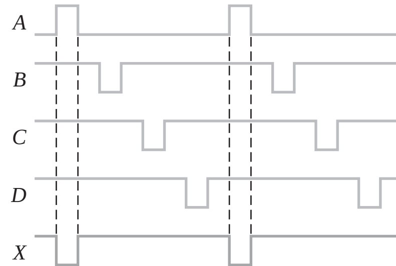


FIGURA 3.102

3.14 *Véase* la Figura 3.103.

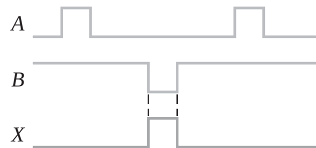


FIGURA 3.103

3.15 *Véase* la Figura 3.104.

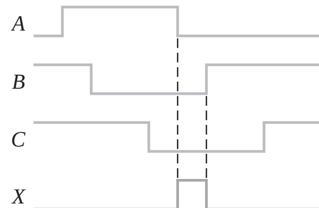


FIGURA 3.104

3.16 Utilizar una puerta NOR de 2 entradas.

3.17 Una puerta NAND de 3 entradas.

3.18 La salida siempre está a nivel BAJO. El cronograma es una línea recta.

3.19 La puerta OR–exclusiva no detectará simultáneamente fallos si ambos circuitos generan la misma salida.

3.20 Las salidas no se ven afectadas.

3.21 6 columnas, 9 filas y tres puertas AND con tres entradas cada una.

3.22 La puerta con t_{PLH} y t_{PHL} igual a 4 ns puede funcionar a la frecuencia más alta.

3.23 10 mW

3.24 La salida de la puerta o el pin 13 de entrada están internamente en circuito abierto.

3.25 El display mostrará una lectura errónea porque el contador continúa hasta que se pone a cero.

3.26 El impulso de habilitación es demasiado corto o el contador se ha puesto a cero demasiado pronto.

AUTOTEST

1. (d) **2.** (d) **3.** (a) **4.** (e) **5.** (c) **6.** (a) **7.** (d)
8. (b) **9.** (d) **10.** (b) **11.** (d) **12.** (c) **13.**(b)
14. (a) **15.** (d) **16.** (c) **17.** (c)

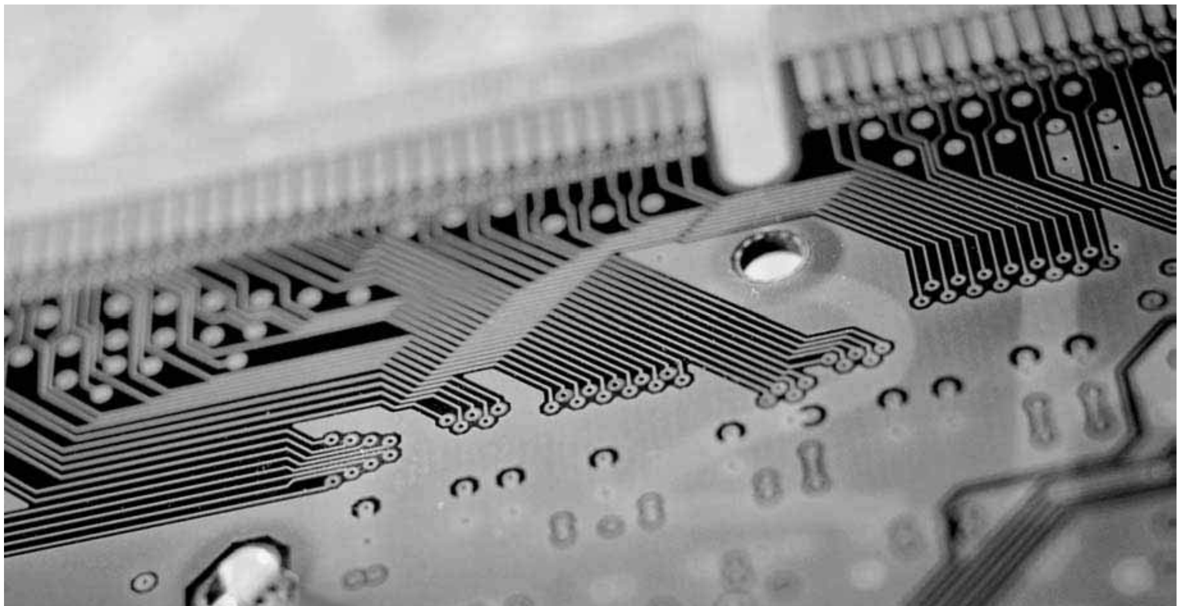
4

ÁLGEBRA DE BOOLE Y SIMPLIFICACIÓN LÓGICA

CONTENIDO DEL CAPÍTULO

OBJETIVOS DEL CAPÍTULO

- Aplicar las leyes y reglas básicas del álgebra de Boole.
- Aplicar los teoremas de DeMorgan a las expresiones booleanas.
- Describir redes de puertas mediante expresiones booleanas.



- Evaluar las expresiones booleanas.
- Simplificar expresiones mediante las leyes y reglas del álgebra booleana.
- Convertir cualquier expresión booleana en una suma de productos.
- Convertir cualquier expresión booleana en un producto de sumas.
- Utilizar el mapa de Karnaugh para simplificar expresiones booleanas.
- Utilizar el mapa de Karnaugh para simplificar tablas de verdad.
- Utilizar condiciones "indiferentes" para simplificar funciones lógicas.
- Aplicar el álgebra de Boole, los mapas de Karnaugh y el lenguaje VHDL a los sistemas digitales.

PALABRAS CLAVE

- Variable
- Complemento
- Término suma
- Término producto
- Suma de productos
- Producto de sumas
- Mapa de Karnaugh
- Indiferente
- VHDL

INTRODUCCIÓN

En 1854, George Boole publicó una obra titulada *Investigación de las leyes del pensamiento, sobre las que se basan las teorías matemáticas de la lógica y la probabilidad*. En esta publicación se formuló la idea de un "álgebra lógica", que se conoce hoy en día como álgebra de Boole. El álgebra de Boole es una forma adecuada y sistemática de expresar y analizar las operaciones de los circuitos lógicos. Claude Shannon fue el primero en aplicar la obra de Boole al análisis y diseño de circuitos. En 1938, Shannon escribió su tesis doctoral en el MIT (*Massachussets Institute of Technology*) titulada *Análisis simbólico de los circuitos de conmutación y relés*.

Este capítulo se ocupa de las leyes, reglas y teoremas del álgebra booleana y sus aplicaciones a los circuitos digitales. Aprenderá a definir un circuito mediante una expresión booleana y a determinar su funcionamiento. También se tratará la simplificación de los circuitos lógicos utilizando el álgebra booleana y los mapas de Karnaugh.

También se presenta el lenguaje de descripción hardware VHDL para la programación de dispositivos lógicos.

APLICACIÓN A LOS SISTEMAS DIGITALES

Esta aplicación a los sistemas digitales ilustra los conceptos que serán explicados a lo largo del capítulo. El funcionamiento del display de 7 segmentos del sistema de control y recuento de pastillas del Capítulo 1 es un buen método para ilustrar la aplicación del álgebra de Boole y de los mapas de Karnaugh, de modo que se obtenga la más sencilla implementación en el diseño de circuitos lógicos. Por tanto, en está aplicación a los sistemas digitales, nos centraremos en la lógica del convertidor BCD-7 segmentos que gobierna los dos displays del sistema indicados en la Figura 1.58.

4.1 OPERACIONES Y EXPRESIONES BOOLEANAS

El álgebra de Boole son las matemáticas de los sistemas digitales. Es indispensable tener unos conocimientos básicos del álgebra booleana para estudiar y analizar los circuitos lógicos. En el capítulo anterior, se han presentado las operaciones y expresiones booleanas para las puertas NOT, AND, OR, NAND y NOR.

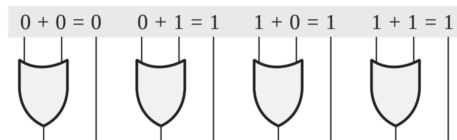
Al finalizar esta sección, el lector deberá ser capaz de:

■ Definir *variable*. ■ Definir *literal*. ■ Identificar un término suma. ■ Evaluar un término suma. ■ Identificar un término producto. ■ Evaluar un término producto. ■ Explicar la adición booleana. ■ Explicar la multiplicación booleana.

Los términos *variable, complemento* y *literal* son términos utilizados en el álgebra booleana. Una ***variable*** es un símbolo (normalmente una letra mayúscula en cursiva) que se utiliza para representar magnitudes lógicas. Cualquier variable puede tener un valor de 0 o de 1. El **complemento** es el inverso de la variable y se indica mediante una barra encima de la misma. Por ejemplo, el complemento de la variable A es $\overline{A}$. Si $A = 1$, entonces $\overline{A} = 0$. Si $A = 0$, entonces $\overline{A} = 1$. El complemento de la variable A se lee "no A" o "A barra". En ocasiones, se emplea un apóstrofe en lugar de la barra para indicar el complemento de una variable; por ejemplo B' indica el complemento de B. En este libro, sólo se utiliza la barra. Un **literal** es una variable o el complemento de una variable.

Suma booleana

Como hemos visto en el Capítulo 3, la **suma booleana** es equivalente a la operación OR y a continuación se muestran sus reglas básicas junto con su relación con la puerta OR:



En el álgebra de Boole, un ***término suma*** es una suma de literales. En los circuitos lógicos, un término suma se obtiene mediante una operación OR, sin que exista ninguna operación AND en la expresión. Algunos ejemplos de términos suma son $A + B$, $A + \overline{B}$, $A + B + \overline{C}$ y $A + B + C + \overline{D}$.

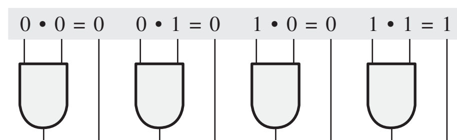
▲ *La puerta OR es un sumador booleano.*

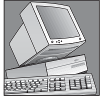
Un término suma es igual a 1 cuando uno o más de los literales del término es 1. Un término suma es igual a 0 sólo si cada uno de los literales son iguales a 0.

Multiplicación booleana

▲ *La puerta AND es un multiplicador booleano.*

En el Capítulo 3 vimos también que la **multiplicación booleana** es equivalente a la operación AND y sus reglas básicas junto con sus relaciones con la puerta AND se ilustran a continuación:





NOTAS INFORMÁTICAS

En un microprocesador, la unidad aritmético lógica (ALU) realiza las operaciones aritméticas y lógicas booleanas sobre los datos digitales mediante instrucciones de programa. Las operaciones lógicas son equivalentes a las operaciones de las puertas básicas con las que ya estamos familiarizados, aunque se trabaja con ocho bits como mínimo a la vez. Ejemplos de instrucciones lógicas booleanas son AND, OR, NOT y XOR, que se denominan *mnemónicos*. Un programa en lenguaje ensamblador utiliza estos mnemónicos para especificar una operación. Y otro programa denominado *ensamblador* traduce los mnemónicos a un código binario que puede entender el microprocesador.

En el álgebra de Boole, un ***término producto*** es un producto de literales. En los circuitos lógicos, un término suma se obtiene mediante una operación AND, sin que existe ninguna operación OR en la expresión. Algunos ejemplos de términos suma son AB, $A\overline{B}, ABC$ y $A\overline{B}C\overline{D}$.

Un término producto es igual a 1 sólo si cada uno de los literales del término es 1. Un término producto es igual a 0 cuando uno o más de los literales son iguales a 0.

EJEMPLO 4.1

Determinar los valores de *A, B, C* y *D* que hacen que el término suma $A+\overline{B}+C+\overline{D}$ sea igual a cero.

Solución Para que el término suma sea 0, cada uno de los literales del término debe ser igual a 0. Por tanto, A = **0**, B = 1 (para que $\overline{B}$ = 0) y D = **1**) para que $\overline{D}$ = 0).

$$A+\overline{B}+C+\overline{D}=0+\overline{1}+0+\overline{1}=0+0+0+0=0$$

Problema relacionado* Determinar los valores de A y B de modo que el término suma $\overline{A}+B$ sea igual a 0.

* Las respuestas se encuentran al final del capítulo.

EJEMPLO 4.2

Determinar los valores de *A, B, C* y *D* que hacen que el término producto $A\overline{B}C\overline{D}$ sea igual a 1.

Solución Para que el término producto sea 1, cada uno de los literales del término debe ser igual a 1. Por tanto, A = **1**, B = **0** (para que $\overline{B}$ = 1), C = **1** y D = **0** (para que $\overline{D}$ = 1).

$$A\overline{B}C\overline{D}=1\cdot\overline{0}\cdot1\cdot\overline{0}=1\cdot1\cdot1\cdot1=1$$

Problema relacionado Determinar los valores de A y B de modo que el término suma $\overline{A}\overline{B}$ sea igual a 1.

REVISIÓN DE LA SECCIÓN 4.1

Las respuestas se encuentran al final del capítulo

1. Si A = 0, ¿cuánto vale $\overline{A}$?
2. Determinar los valores de A, B y C que hacen que el término suma $\overline{A}+\overline{B}+C$ sea igual a 0.
3. Determinar los valores de A, B y C que hacen que el término producto $A\overline{B}C$ sea igual a 1.

4.2 LEYES Y REGLAS DEL ÁLGEBRA DE BOOLE

Al igual que en otras áreas de las matemáticas, existen en el álgebra de Boole una serie de reglas y leyes bien determinadas que tienen que seguirse para aplicarla correctamente. Las más importantes son las que se presentan en esta sección.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Aplicar las leyes conmutativas de la adición y multiplicación. ■ Aplicar las leyes asociativas de la adición y multiplicación. ■ Aplicar la ley distributiva. ■ Aplicar las doce reglas básicas del álgebra de Boole.

Leyes del álgebra de Boole

Las leyes básicas del álgebra de Boole (las **leyes conmutativas** de la suma y la multiplicación, y las **leyes asociativas** de la suma y la multiplicación y la **ley distributiva**) son las mismas que las del álgebra ordinaria. Cada una de las leyes se ilustra con dos o tres variables, pero el número de variables no está limitado a esta cantidad.

Leyes conmutativas La *ley conmutativa de la suma* para dos variables se escribe como sigue:

Ecuación 4.1 $$A + B = B + A$$

Esta ley establece que el orden en que se aplica a las variables la operación OR es indiferente. Recuerde que cuando se aplica a los circuitos lógicos, la suma y la operación OR es lo mismo. La Figura 4.1 ilustra la ley conmutativa aplicada a una puerta OR, en la que se puede ver que es indistinto a qué entrada asignemos cada una de las variables. (El símbolo ≡ significa “equivalente a”.)

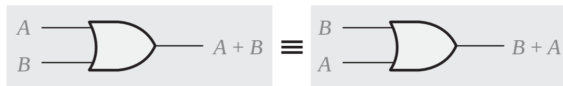


FIGURA 4.1 Aplicación de la ley conmutativa de la suma.

La *ley conmutativa de la multiplicación* para dos variables es

Ecuación 4.2 $$AB = BA$$

Esta ley establece que el orden en que se aplica a las variables la operación AND es indiferente. La Figura 4.2 ilustra esta ley tal y como se aplica a la puerta AND.

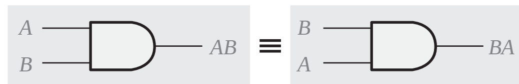


FIGURA 4.2 Aplicación de la ley conmutativa de la multiplicación.

Leyes asociativas La *ley asociativa de la suma* para tres variables se escribe como sigue:

Ecuación 4.3 $$A + (B + C) = (A + B) + C$$

Esta ley establece que cuando se aplica la operación OR a más de dos variables, el resultado es el mismo independientemente de la forma en que se agrupen las variables. La Figura 4.3 ilustra esta ley aplicada a puertas OR de dos entradas.

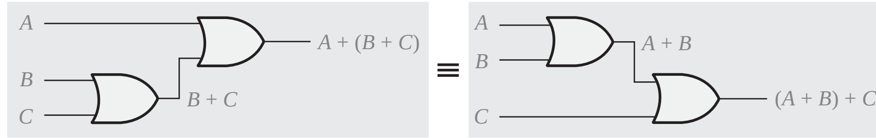


FIGURA 4.3 Aplicación de la ley asociativa de la suma.

La *ley asociativa de la multiplicación* para tres variables se escribe del siguiente modo:

Ecuación 4.4 $A(BC) = (AB)C$

Esta ley establece que cuando se aplica la operación AND a más de dos variables, el resultado es el mismo independientemente de la forma en que se agrupen las variables. La Figura 4.4 ilustra esta ley aplicada a puertas AND de dos entradas.

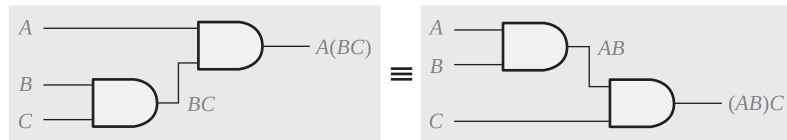


FIGURA 4.4 Aplicación de la ley asociativa de la multiplicación.

Ley distributiva La *ley distributiva* para tres variables se escribe como sigue:

Ecuación 4.5 $A(B + C) = AB + AC$

Esta ley establece que aplicar la operación OR a dos o más variables y luego aplicar la operación AND al resultado de esa operación y a otra variable aislada, es equivalente a aplicar la operación AND a la variable aislada con cada uno de los sumandos y luego realizar la operación OR con los productos resultantes. La ley distributiva expresa también el proceso de *sacar factor común* en el que la variable común A se saca como factor de los productos parciales, como por ejemplo, $AB + AC = A(B + C)$. La Figura 4.5 ilustra la ley distributiva mediante su implementación de puertas.

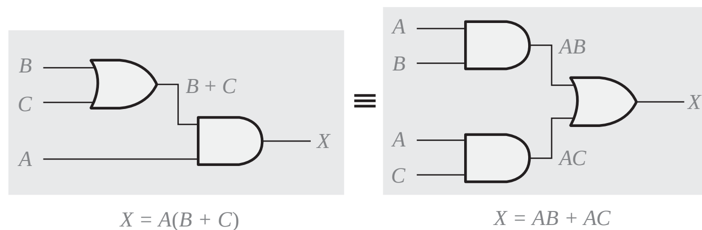


FIGURA 4.5 Aplicación de la ley distributiva.

Reglas del álgebra booleana

La Tabla 4.1 enumera las doce reglas básicas, muy útiles, para la manipulación y simplificación de **expresiones booleanas**. Las nueve primeras reglas las veremos en términos de su aplicación a las puertas lógicas. Las reglas 10 a 12 se obtendrán a partir de las reglas más sencillas y de las leyes anteriormente explicadas.

1. $A + 0 = A$	**7.** $A \cdot A = A$
2. $A + 1 = 1$	**8.** $A \cdot \overline{A} = 0$
3. $A \cdot 0 = 0$	**9.** $\overline{\overline{A}} = A$
4. $A \cdot 1 = A$	**10.** $A + AB = A$
5. $A + A = A$	**11.** $A + \overline{A}B = A + B$
6. $A + \overline{A} = 1$	**12.** $(A + B)(A + C) = A + BC$

A, *B* o *C* pueden representar una sola variable o una combinación de variables.

TABLA 4.1 Reglas básicas del Álgebra de Boole.

Regla 1. $A + 0 = A$ Si aplicamos la operación OR a una variable cualquiera y a 0, el resultado es siempre igual a la variable. Si A es 1, la salida es igual a 1 y, por tanto, igual a A. Si A es 0, la salida es 0 e igualmente idéntica a A. Esta ley se ilustra en la Figura 4.6 en la que la entrada inferior está siempre a 0.

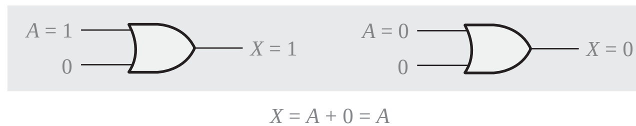


FIGURA 4.6

Regla 2. $A + 1 = 1$ Si se aplica la operación OR a una variable y a 1, el resultado es siempre igual a 1. Un 1 en una entrada de una puerta OR produce siempre un 1 en la salida, independientemente del valor de la otra entrada. Esta regla se ilustra en la Figura 4.7, en la que la entrada inferior está siempre a 1.

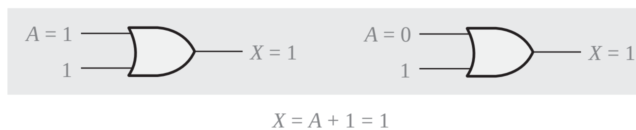


FIGURA 4.7

Regla 3. $A \cdot 0 = 0$ Si se aplica la operación AND a una variable y a 0, el resultado es siempre igual a 0. Siempre que una de las entradas de una puerta AND sea 0, la salida siempre es 0, independientemente del valor de la otra entrada. Esta regla se ilustra en la Figura 4.8, en la que la entrada inferior está siempre a 0.

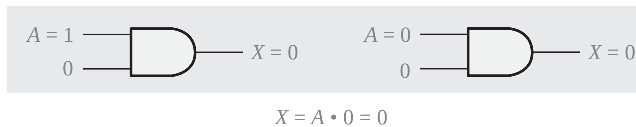


FIGURA 4.8

Regla 4. $A \cdot 1 = A$ Si se aplica la operación AND a una variable y a 1, el resultado es siempre igual a la variable. Si la variable A es 0, la salida de la puerta AND será siempre 0, mientras que si A es 1, la salida será 1, dado que las dos entradas son 1. Esta regla se ilustra en la Figura 4.9, en la que la entrada inferior está siempre a 1.

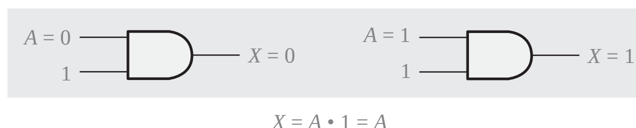


FIGURA 4.9

Regla 5. $A + A = A$ Si se aplica la operación OR a una variable consigo misma, el resultado es siempre igual a la variable. Si *A* es 0, entonces 0 + 0 = 0, mientras que si *A* es 1, 1 + 1 =1. Esto se muestra en la Figura 4.10, en la que se aplica la misma variable a ambas entradas.

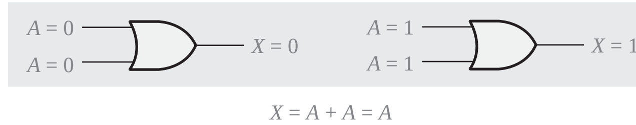


FIGURA 4.10

Regla 6. $A + \overline{A} = 1$ Si se aplica la operación OR a una variable y a su complemento, el resultado es siempre igual a 1. Si *A* es 0, entonces $0 + \overline{0} = 0 + 1 = 1$. Si *A* es 1, entonces $1 + \overline{1} = 1 + 0 = 1$. En la Figura 4.11 podemos ver una puerta OR en la que sus entradas son una variable y su complemento.

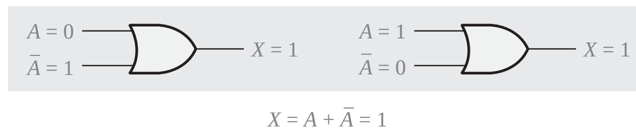


FIGURA 4.11

Regla 7. $A \cdot A = A$ Si se aplica la operación AND a una variable consigo misma, el resultado siempre es igual a la variable. Si A = 0, entonces 0 · 0 = 0, y si *A* = 1, entonces 1 · 1 = 1. Esta regla se ilustra en la Figura 4.12.

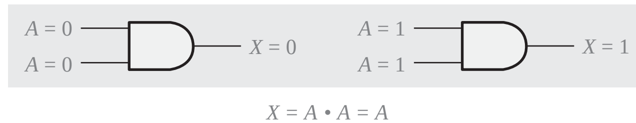


FIGURA 4.12

Regla 8. $A \cdot \overline{A} = 0$ Si se aplica la operación AND a una variable y a su complemento, el resultado es siempre igual a 0. Esta regla se basa en que siempre *A* o $\overline{A}$ será 0, y además en que cuando se aplica un 0 a una de las entradas de una puerta AND, la salida siempre es 0. Esta regla se ilustra en la Figura 4.13.

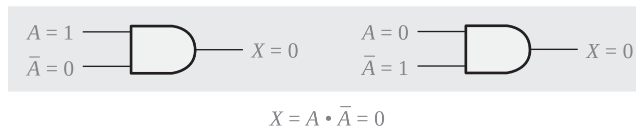


FIGURA 4.13

Regla 9. $\overline{\overline{A}} = A$ El complemento del complemento de una variable es siempre la propia variable. El complemento de la variable *A* es $\overline{A}$ y el complemento de $\overline{A}$ será de nuevo *A*, que es la variable original. Esta regla se muestra en la Figura 4.14 mediante el uso de dos inversores.

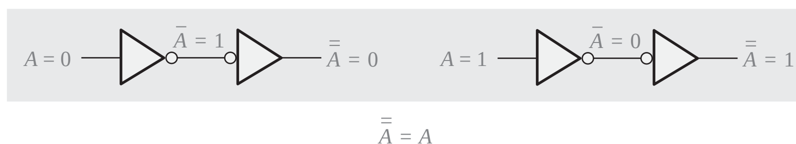


FIGURA 4.14

Regla 10. $A + AB = A$ Esta regla se puede obtener aplicando la ley distributiva y las reglas 2 y 4, de la siguiente forma:

$A + AB = A(1 + B)$ Sacar factor común (ley distributiva)
$= A \cdot 1$ Regla 2: $(1 + B) = 1$
$= A$ Regla 4: $A \cdot 1 = A$

La demostración se muestra en la Tabla 4.2, la cual incluye la tabla de verdad y la simplificación del circuito lógico resultante.

A	B	AB	$A + AB$
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	1

igual

A
B
A
conexión directa

TABLA 4.2 Regla 10: $A + AB = A$.

Regla 11. $A + \overline{A}B = A + B$ Esta regla puede demostrarse de la siguiente forma:

$A + \overline{A}B = (A + AB) + \overline{A}B$ Regla 10: $A = A + AB$
$= (AA + AB) + \overline{A}B$ Regla 7: $A = AA$
$= AA + AB + A\overline{A} + \overline{A}B$ Regla 8: sumar $A\overline{A} = 0$
$= (A + \overline{A})(A + B)$ Sacar factor común
$= 1 \cdot (A + B)$ Regla 6: $A + \overline{A} = 1$
$= A + B$ Regla 4: eliminar el 1

La demostración se muestra en la Tabla 4.3, la cual incluye la tabla de verdad y la simplificación del circuito lógico resultante.

A	B	$\overline{A}B$	$A + \overline{A}B$	$A + B$
0	0	0	0	0
0	1	1	1	1
1	0	0	1	1
1	1	0	1	1

igual

A
B
A
B

TABLA 4.3 Regla 11: $A + \overline{A}B = A + B$.

Regla 12. $(A + B)(A + C) = A + BC$ Esta regla puede demostrarse de la siguiente forma:

$(A + B)(A + C) = AA + AC + AB + BC$ Ley distributiva
$= A + AC + AB + BC$ Regla 7: $AA = A$
$= A(1 + C) + AB + BC$ Sacar factor común (ley distributiva)

$$= A \cdot 1 + AB + BC \qquad \text{Regla 2: } 1 + C = 1$$
$$= A(1 + B) + BC \qquad \text{Sacar factor común (ley distributiva)}$$
$$= A \cdot 1 + BC \qquad \text{Regla 2: } 1 + B = 1$$
$$= A + BC \qquad \text{Regla 4: } A \cdot 1 = A$$

La demostración se muestra en la Tabla 4.4, la cual incluye la tabla de verdad y la simplificación del circuito lógico resultante.

A	B	C	$A+B$	$A+C$	$(A+B)(A+C)$	BC	$A+BC$
0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	1	1	1	1
1	0	0	1	1	1	0	1
1	0	1	1	1	1	0	1
1	1	0	1	1	1	0	1
1	1	1	1	1	1	1	1

igual

TABLA 4.4 Regla 12: $(A + B)(A + C) = A + BC$.

REVISIÓN DE LA SECCIÓN 4.2

1. Aplicar la ley asociativa de la adición a la expresión $A+(B+C+D)$.
2. Aplicar la ley distributiva a la expresión $A(B+C+D)$.

4.3 TEOREMAS DE DeMORGAN

DeMorgan, matemático que conoció a Boole, propuso dos teoremas que constituyen una parte muy importante del álgebra de Boole. En términos prácticos, los teoremas de DeMorgan proporcionan una verificación matemática de la equivalencia entre las puertas NAND y negativa-OR, y las puertas NOR y negativa-AND, que se han tratado en el Capítulo 3.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Enunciar los teoremas de DeMorgan. ■ Relacionar los teoremas de DeMorgan con la equivalencia entre las puertas NAND y negativa-OR, y entre las puertas NOR y negativa-AND. ■ Aplicar los teoremas de DeMorgan para simplificar las expresiones booleanas.

El primer teorema de DeMorgan se enuncia de la siguiente forma:

El complemento de un producto de variables es igual a la suma de los complementos de las variables.

O dicho de otra manera

El complemento de dos o más variables a las que se aplica la operación AND es equivalente a aplicar la operación OR a los complementos de cada variable.

La fórmula para expresar este teorema para dos variables es:

Ecuación 4.6 $$\overline{XY} = \overline{X} + \overline{Y}$$

El segundo teorema de DeMorgan se enuncia como sigue:

El complemento de una suma de variables es igual al producto de los complementos de las variables.

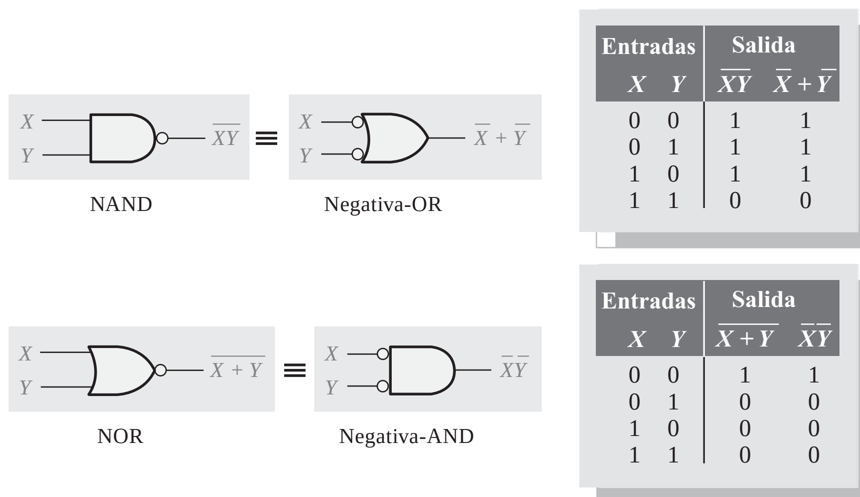
O dicho de otra manera,

El complemento de dos o más variables a las que se aplica la operación OR es equivalente a aplicar la operación AND a los complementos de cada variable.

La fórmula para expresar este teorema es:

Ecuación 4.7 $$\overline{X + Y} = \overline{X}\,\overline{Y}$$

Las puertas equivalentes y tablas de verdad correspondientes a las Ecuaciones 4.6 y 4.7 se muestran en la Figura 4.15.



Entradas		Salida	
X	Y	$\overline{XY}$	$\overline{X}+\overline{Y}$
0	0	1	1
0	1	1	1
1	0	1	1
1	1	0	0

Entradas		Salida	
X	Y	$\overline{X+Y}$	$\overline{X}\,\overline{Y}$
0	0	1	1
0	1	0	0
1	0	0	0
1	1	0	0

FIGURA 4.15 Equivalencias de las puertas lógicas y tablas de verdad que ilustran los teoremas de DeMorgan. Observe la igualdad entre las dos columnas de salida de cada tabla. Esto demuestra que las puertas equivalentes realizan la misma función lógica.

Como se ha comentado, los teoremas de DeMorgan se aplican también a expresiones en las que existen más de dos variables. Los siguientes ejemplos ilustran la aplicación de los teoremas de DeMorgan a expresiones de 3 y 4 variables.

EJEMPLO 4.3

Aplicar los teoremas de DeMorgan a las expresiones $\overline{XYZ}$ y $\overline{X+Y+Z}$.

$$\overline{XYZ} = \overline{X} + \overline{Y} + \overline{Z}$$

$$\overline{X+Y+Z} = \overline{X}\,\overline{Y}\,\overline{Z}$$

Problema relacionado Aplicar los teoremas de DeMorgan a la expresión $\overline{\overline{X}+\overline{Y}+\overline{Z}}$..

EJEMPLO 4.4

Aplicar los teoremas de DeMorgan a las expresiones $\overline{WXYZ}$ y $\overline{W+X+Y+Z}$.

$$\overline{WXYZ} = \overline{W}+\overline{X}+\overline{Y}+\overline{Z}$$
$$\overline{W+X+Y+Z} = \overline{W}\,\overline{X}\,\overline{Y}\,\overline{Z}$$

Problema relacionado Aplicar los teoremas de DeMorgan a la expresión $\overline{\overline{W}\,\overline{X}\,\overline{Y}\,\overline{Z}}$

Como se ha establecido en las Ecuaciones 4.6 y 4.7 que enuncian los teoremas de DeMorgan, cada variable puede representar una combinación de otras variables. Por ejemplo, X puede ser igual al término $AB + C$, e Y puede ser igual a $A + BC$. De esta forma, si aplicamos el teorema de DeMorgan para dos variables expresado según $\overline{XY} = \overline{X}+\overline{Y}$ a la expresión $\overline{(AB+C)(A+BC)}$ obtenemos el siguiente resultado:

$$\overline{(AB+C)(A+BC)} = (\overline{AB+C})+(\overline{A+BC})$$

Observe que el resultado anterior tiene dos términos $\overline{AB+C}$ y $\overline{A+BC}$, a los que podemos aplicar individualmente otra vez el teorema de DeMorgan $\overline{X+Y} = \overline{X}\,\overline{Y}$ del siguiente modo:

$$(\overline{AB+C})+(\overline{A+BC}) = (\overline{AB})\overline{C}+\overline{A}(\overline{BC})$$

De esta manera obtenemos otros dos términos en la expresión a los que de nuevo podemos aplicar el teorema de DeMorgan. Estos términos son $\overline{AB}$ y $\overline{BC}$. Una última aplicación del teorema de DeMorgan nos proporciona el siguiente resultado:

$$(\overline{AB})\overline{C}+\overline{A}(\overline{BC}) = (\overline{A}+\overline{B})\overline{C}+\overline{A}(\overline{B}+\overline{C})$$

Aunque este resultado puede simplificarse aún más utilizando las leyes y reglas de Boole, los teoremas de DeMorgan ya no se pueden aplicar más.

Aplicación de los teoremas de DeMorgan

El siguiente procedimiento ilustra la aplicación de los teoremas de DeMorgan y del álgebra de Boole a la expresión:

$$\overline{\overline{A+B\overline{\overline{C}}}+D(\overline{E+\overline{\overline{F}}})}$$

Paso 1. Identificamos los términos a los que se pueden aplicar los teorema de DeMorgan y consideramos cada término como una única variable, por lo que establecemos $\overline{A+B\overline{\overline{C}}} = X$ y $D(\overline{E+\overline{\overline{F}}}) = Y$.

Paso 2. Dado que $\overline{X+Y} = \overline{X}\,\overline{Y}$.

$$\overline{(\overline{A+B\overline{\overline{C}}})+(\overline{D(E+\overline{\overline{F}})})} = (\overline{\overline{A+B\overline{\overline{C}}}})(\overline{D(\overline{E+\overline{\overline{F}}})})$$

Paso 3. Utilizamos la regla 9 $(\overline{\overline{A}} = A)$ para eliminar la bara doble sobre el término de la izquierda (esto no es parte del teorema de DeMorgan).

$$(\overline{\overline{A+B\overline{\overline{C}}}})(\overline{D(\overline{E+\overline{\overline{F}}})}) = (A+B\overline{C})(\overline{D(\overline{E+\overline{\overline{F}}})})$$

Paso 4. Aplicando el teorema de DeMorgan al segundo término:

$$(A+B\overline{C})(\overline{D(\overline{E+\overline{F}})})=(A+B\overline{C})(\overline{D}+(\overline{\overline{E+\overline{F}}}))$$

Paso 5. Empleamos la regla 9 ($\overline{\overline{A}}=A$) para cancelar las barras dobles sobre la parte $E+\overline{F}$ del término.

$$(A+B\overline{C})(\overline{D}+(\overline{\overline{E+\overline{F}}}))=(A+B\overline{C})(\overline{D}+E+\overline{F})$$

Los siguientes tres ejemplos ilustrarán detalladamente cómo emplear los teoremas de DeMorgan.

EJEMPLO 4.5

Aplicar los teoremas de DeMorgan a cada una de las siguientes expresiones:

(a) $\overline{(A+B+C)D}$ **(b)** $\overline{ABC+DEF}$ **(c)** $\overline{A\overline{B}+\overline{C}D+EF}$

Solución

(a) Sea $A+B+C=X$ y $D=Y$. La expresión $\overline{(A+B+C)D}$ es de la forma $\overline{XY}=\overline{X}+\overline{Y}$ y se puede escribir como sigue:

$$\overline{(A+B+C)D}=\overline{A+B+C}+\overline{D}$$

A continuación, aplicamos el teorema de DeMorgan al término $\overline{A+B+C}$

$$\overline{A+B+C}+\overline{D}=\overline{A}\overline{B}\overline{C}+\overline{D}$$

(b) Sea $ABC=X$ y $DEF=Y$. La expresión $\overline{ABC+DEF}$ es de la forma $\overline{X+Y}=\overline{X}\overline{Y}$ y podemos reescribirla de la forma:

$$\overline{ABC+DEF}=(\overline{ABC})(\overline{DEF})$$

A continuación, aplicamos el teorema de DeMorgan a cada uno de los términos $\overline{ABC}$ y $\overline{DEF}$.

$$(\overline{ABC})(\overline{DEF})=(\overline{A}+\overline{B}+\overline{C})(\overline{D}+\overline{E}+\overline{F})$$

(c) Sean $A\overline{B}=X, \overline{C}D=Y$ y $EF=Z$. La expresión $\overline{A\overline{B}+\overline{C}D+EF}$ es de la forma $\overline{X+Y+Z}=\overline{X}\overline{Y}\,\overline{Z}$ y se puede reescribir como:

$$\overline{A\overline{B}+\overline{C}D+EF}=(\overline{A\overline{B}})(\overline{\overline{C}D})(\overline{EF})$$

A continuación, aplicamos el teorema de DeMorgan a cada uno de los términos $\overline{A\overline{B}}, \overline{\overline{C}D}$ y $\overline{EF}$.

$$(\overline{A\overline{B}})(\overline{\overline{C}D})(\overline{EF})=(\overline{A}+B)(C+\overline{D})(\overline{E}+\overline{F})$$

Problema relacionado Aplicar los teoremas de DeMorgan a la expresión $\overline{\overline{ABC}+D+E}$.

EJEMPLO 4.6

Aplicar los teoremas de DeMorgan a cada una de las siguientes expresiones:

(a) $\overline{(\overline{A+B})+\overline{C}}$ **(b)** $\overline{(\overline{A}+B)+CD}$ **(c)** $\overline{(A+B)\overline{C}\,\overline{D}+E+\overline{F}}$

Solución

(a) $\overline{(\overline{A+B})+\overline{C}} = (\overline{\overline{A+B}})\overline{\overline{C}} = (A+B)C$

(b) $\overline{(\overline{A}+B)+CD} = (\overline{\overline{A}+B})\overline{CD} = (\overline{\overline{A}}\,\overline{B})(\overline{C}+\overline{D}) = A\overline{B}(\overline{C}+\overline{D})$

(c) $\overline{(A+B)\overline{C}\,\overline{D}+E+\overline{F}} = ((\overline{(A+B)\overline{C}\,\overline{D}})(\overline{E+\overline{F}}) = (\overline{A}\,\overline{B}+C+D)\overline{E}F$

Problema relacionado Aplicar los teoremas de DeMorgan a la expresión $\overline{\overline{A}B(C+\overline{D})+E}$.

EJEMPLO 4.7

La expresión booleana de una puerta OR-exclusiva es $A\overline{B}+\overline{A}B$. Tomando esto como punto de partida, desarrollar una expresión para una puerta NOR-exclusiva, utilizando los teoremas de DeMorgan y aquellas leyes o reglas que puedan aplicarse.

Solución En primer lugar se complementa la expresión OR-exclusiva y luego se aplican los teorema de DeMorgan del siguiente modo:

$$\overline{A\overline{B}+\overline{A}B} = (\overline{A\overline{B}})(\overline{\overline{A}B}) = (\overline{A}+\overline{\overline{B}})(\overline{\overline{A}}+\overline{B}) = (\overline{A}+B)(A+\overline{B})$$

A continuación se aplica la ley distributiva y la regla 8 $(A\cdot\overline{A}=0)$.

$$(\overline{A}+B)(A+\overline{B}) = \overline{A}A+\overline{A}\,\overline{B}+AB+B\overline{B} = \overline{A}\,\overline{B}+AB$$

La expresión resultante para una puerta XNOR es $\overline{A}\,\overline{B}+AB$. Observe que esta expresión es igual a 1 siempre que ambas variables sean 0 o 1.

Problema relacionado A partir de la expresión para una puerta NAND de 4 entradas, utilizar los teoremas de DeMorgan para desarrollar una expresión para una puerta negativa-OR de 4 entradas.

REVISIÓN DE LA SECCIÓN 4.3

1. Aplicar los teoremas de DeMorgan a las siguientes expresiones:

(a) $\overline{ABC}+(\overline{\overline{D}+E})$ **(b)** $\overline{(A+B)C}$ **(c)** $\overline{A+B+C}+\overline{\overline{D}E}$

4.4 ANÁLISIS BOOLEANO DE LOS CIRCUITOS LÓGICOS

El álgebra de Boole proporciona una manera concisa de expresar el funcionamiento de un circuito lógico formado por una combinación de puertas lógicas, de tal forma que la salida puede determinarse por la combinación de los valores de entrada.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Determinar las expresiones booleanas de una combinación de puertas. ■ Evaluar el funcionamiento lógico de un circuito a partir de su expresión booleana. ■ Construir una tabla de verdad.

Expresión booleana de un circuito lógico

▲ *Un circuito lógico se puede describir mediante una ecuación booleana.*

Para obtener la expresión booleana de un determinado circuito lógico, la manera de proceder consiste en comenzar con las entradas situadas más a la izquierda e ir avanzando hasta las líneas de salida, escribiendo la expresión para cada puerta. Para el circuito ejemplo de la Figura 4.16, su expresión booleana se determina de la siguiente manera:

1. La expresión de la puerta AND situada más a la izquierda cuyas entradas son C y D es CD.
2. La salida de la puerta AND situada más a la izquierda es una de las entradas de la puerta OR y B es su otra entrada. Por tanto, la expresión para la puerta OR es $B+CD$.
3. La salida de la puerta OR es una de las entradas de la puerta AND situada más a la derecha, siendo A su otra entrada. Por tanto, la expresión de esta puerta AND será $A(B+CD)$, que es la expresión final de salida del circuito completo.

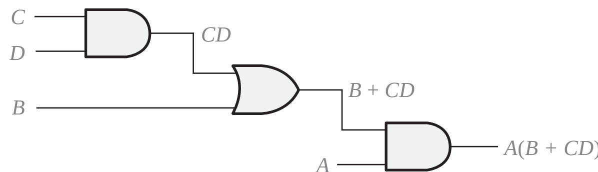


FIGURA 4.16 Circuito lógico que muestra el desarrollo de la expresión booleana para la salida.

Construcción de una tabla de verdad para un circuito lógico

▲ *Un circuito lógico puede describirse mediante una tabla de verdad.*

Una vez que se ha determinado la expresión booleana de un circuito dado, puede desarrollarse una tabla de verdad que represente la salida del circuito lógico para todos los valores posibles de las variables de entrada. El procedimiento requiere que se evalúe la expresión booleana para todas las posibles combinaciones de valores de las variables de entrada. En el caso del circuito de la Figura 4.16, existen cuatro variables de entrada (A, B, C y D) y, por tanto, hay dieciséis ($2^4 = 16$) posibles combinaciones de valores.

Evaluación de la expresión. Para evaluar la expresión $A(B + CD)$, en primer lugar hallamos los valores de las variables que hacen que la expresión sea igual a 1, utilizando las reglas de la suma y la multiplicación booleanas. En este caso, la expresión es igual a 1 sólo si $A = 1$ y $B + CD = 1$, ya que:

$$A(B + CD) = 1 \cdot 1 = 1$$

Ahora hay que determinar cuándo el término $B + CD$ es igual a 1. El término $B + CD = 1$ si $B = 1$ o $C = 1$ o si ambas variables son igual a 1, ya que:

$$B + CD = 1 + 0 = 1$$

$$B + CD = 0 + 1 = 1$$

$$B + CD = 1 + 1 = 1$$

El término $CD = 1$ sólo si $C = 1$ y $D = 1$.

Resumiendo, la expresión $A\,(B + CD) = 1$ cuando $A = 1$ y $B = 1$, independientemente de los valores de C y D, o cuando $A = 1$ y $C = 1$ o cuando $A = 1$ y $C = 1$ y $D = 1$, independientemente del valor de B. La expresión $A\,(B + CD) = 0$ para todas las restantes combinaciones de valores de las variables.

Representación de los resultados en una tabla de verdad. El primer paso consiste en enumerar las dieciséis combinaciones de unos y ceros de las variables de entrada en una secuencia binaria, como muestra la Tabla 4.5. A continuación, se pone un 1 en la columna de salida para las combinaciones de variables de entrada que se han determinado en la evaluación de la expresión. Finalmente, se escribe un 0 en la columna de salida para el resto de las combinaciones de las variables de entrada. Estos resultados se muestran en la Tabla 4.5.

Entradas				**Salida**
A	***B***	***C***	***D***	***A(B + CD)***
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

TABLA 4.5 Tabla de verdad del circuito lógico de la Figura 4.16.

REVISIÓN DE LA SECCIÓN 4.4

1. Reemplazar las puertas AND por puertas OR y la puerta OR por una puerta AND en la Figura 4.16, y determinar la expresión booleana de salida.
2. Elaborar la tabla de verdad del circuito de la cuestión 1.

4.5 SIMPLIFICACIÓN MEDIANTE EL ÁLGEBRA DE BOOLE

Muchas veces, a la hora de aplicar el álgebra booleana, hay que reducir una expresión a su forma más simple o cambiarla a una forma más conveniente para conseguir una implementación más eficiente. El método que se va a tratar en esta sección utiliza las reglas, leyes y teoremas del álgebra de Boole para manipular y simplificar una expresión. Este método requiere un profundo conocimiento del álgebra booleana y una considerable experiencia en su aplicación, por no mencionar también un poquito de ingenio y destreza.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Aplicar las leyes, reglas y teoremas del álgebra de Boole para simplificar cualquier expresión.

Una expresión booleana simplificada emplea el menor número posible de puertas en la implementación de una determinada expresión. Los Ejemplos 4.8 hasta 4.11 ilustran la simplificación booleana.

EJEMPLO 4.8

Simplificar la siguiente expresión utilizando técnicas del álgebra de Boole:

$$AB + A(B + C) + B(B + C)$$

Solución El método que se sigue no es necesariamente el único método posible.

Paso 1. Aplicar la ley distributiva al segundo y tercer término del siguiente modo:

$$AB + AB + AC + BB + BC$$

Paso 2. Aplicar la regla 7 ($BB = B$) al cuarto término.

$$AB + AB + AC + B + BC$$

Paso 3. Aplicar la regla 5 ($AB + AB = AB$) a los dos primeros términos.

$$AB + AC + B + BC$$

Paso 4. Aplicar la regla 10 ($B + BC = B$) a los dos últimos términos.

$$AB + AC + B$$

Paso 5. Aplicar la regla 10 ($AB + B = B$) al primero y tercer término.

$$B + AC$$

En este punto, la expresión ya no puede seguir simplificándose. Según vaya adquiriendo experiencia en la aplicación del álgebra de Boole, podrá combinar muchos de los pasos individuales.

Problema relacionado Simplificar la expresión booleana $A\overline{B} + A(\overline{B+C}) + B(\overline{B+C})$.

▲ *La simplificación consiste en implementar una función con el menor número de puertas posible.*

La Figura 4.17 muestra cómo el proceso de simplificación del Ejemplo 4.8 ha reducido significativamente el número de puertas lógicas necesarias para implementar la expresión. En la parte (a) se puede ver que son necesarias cinco puertas para implementar dicha expresión en su forma original, mientras que sólo se requieren dos para hacerlo una vez simplificada, como se muestra en la parte (b). Es importante resaltar que estos dos circuitos de puertas son equivalentes, es decir, para cualquier combinación de valores en las entradas A, B y C, obtenemos siempre la misma salida en ambos circuitos.

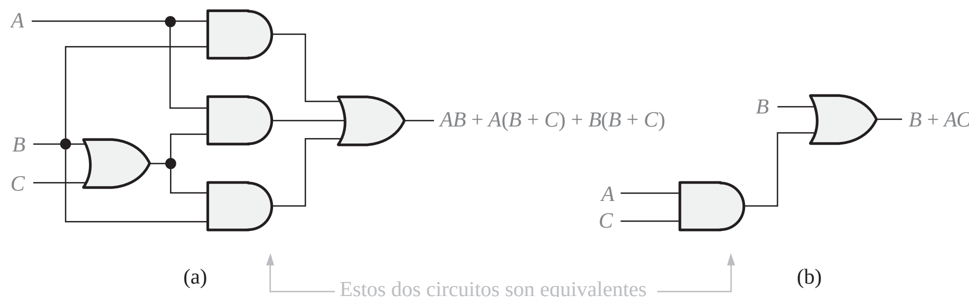


FIGURA 4.17 Circuitos de puertas para el Ejemplo 4.8.

EJEMPLO 4.9

Simplificar la siguiente expresión booleana:

$$[A\overline{B}(C+BD)+\overline{A}\overline{B}]C$$

Tenga en cuenta que los corchetes y paréntesis significan lo mismo: el término en su interior se multiplica (AND) por el término exterior.

Solución **Paso 1.** Aplicar la ley distributiva a los términos entre corchetes.

$$(A\overline{B}C+A\overline{B}BD+\overline{A}\overline{B})C$$

Paso 2. Aplicar la regla 8 ($\overline{B}B=0$) al segundo término entre paréntesis.

$$(A\overline{B}C+A\cdot 0\cdot D+\overline{A}\overline{B})C$$

Paso 3. Aplicar la regla 3 ($A \cdot 0 \cdot D = 0$) al segundo término contenido dentro de los paréntesis.

$$(A\overline{B}C+0+\overline{A}\overline{B})C$$

Paso 4. Aplicar la regla 1 (quitar el 0) dentro del paréntesis

$$(A\overline{B}C+\overline{A}\overline{B})C$$

Paso 5. Aplicar la ley distributiva.

$$A\overline{B}CC+\overline{A}\overline{B}C$$

Paso 6. Aplicar la regla 7 ($CC = C$) al primer término.

$$A\overline{B}C+\overline{A}\overline{B}C$$

Paso 7. Sacar $\overline{B}C$ factor común.

$$\overline{B}C(A+\overline{A})$$

Paso 8. Aplicar la regla 6 ($A+\overline{A}=1$).

$$\overline{B}C\cdot 1$$

Paso 9. Aplicar la regla 4 (quitar el 1).

$$\overline{B}C$$

Problema relacionado Simplificar la expresión booleana $[AB(C+\overline{BD})+\overline{AB}]CD$.

EJEMPLO 4.10

Simplificar la siguiente expresión booleana:

$$\overline{A}BC+A\overline{B}\overline{C}+\overline{A}\overline{B}\overline{C}+A\overline{B}C+ABC$$

Solución **Paso 1.** Sacar factor común BC del primer y último término.

$$BC(\overline{A}+A)+A\overline{B}\overline{C}+\overline{A}\overline{B}\overline{C}+A\overline{B}C$$

Paso 2. Aplicar la regla 6 $(A+\overline{A}=1)$ al término entre paréntesis y sacar factor común $A\overline{B}$ del segundo y último término.

$$BC\cdot 1+A\overline{B}(\overline{C}+C)+\overline{A}\overline{B}\overline{C}$$

Paso 3. Aplicar la regla número 4 (quitar el 1) al primer término y la regla 6 $(\overline{C}+C=1)$ al término entre paréntesis.

$$BC+A\overline{B}\cdot 1+\overline{A}\overline{B}\overline{C}$$

Paso 4. Aplicar la regla 4 (quitar el 1) al segundo término.

$$BC+A\overline{B}+\overline{A}\overline{B}\overline{C}$$

Paso 5. Sacar $\overline{B}$ factor común al segundo y tercer término.

$$BC+\overline{B}(A+\overline{A}\overline{C})$$

Paso 6. Aplicar la regla 11 $(A+\overline{A}\overline{C}=A+\overline{C})$ al término entre paréntesis.

$$BC+\overline{B}(A+\overline{C})$$

Paso 7. Utilizar las leyes distributiva y conmutativa para obtener la siguiente expresión.

$$BC+A\overline{B}+\overline{B}\overline{C}$$

Problema relacionado Simplificar la expresión booleana $AB\overline{C}+\overline{A}\overline{B}C+\overline{A}BC+\overline{A}\overline{B}\overline{C}$.

EJEMPLO 4.11

Simplificar la siguiente expresión booleana:

$$\overline{AB+AC}+\overline{A}\overline{B}C$$

Solución

Paso 1. Aplicar el teorema de DeMorgan al primer término.

$$(\overline{AB})(\overline{AC})+\overline{A}\overline{B}C$$

Paso 2. Aplicar el teorema de DeMorgan a cada uno de los términos entre paréntesis.

$$(\overline{A}+\overline{B})(\overline{A}+\overline{C})+\overline{A}\overline{B}C$$

Paso 3. Aplicar la ley distributiva a los dos términos entre paréntesis.

$$\overline{A}\overline{A}+\overline{A}\overline{C}+\overline{A}\overline{B}+\overline{B}\overline{C}+\overline{A}\overline{B}C$$

Paso 4. Aplicar la regla número 7 $(\overline{A}\overline{A}=\overline{A})$ al primer término y la regla 10 $[\overline{A}\overline{B}+\overline{A}\overline{B}C=\overline{A}\overline{B}(1+C)=\overline{A}\overline{B}]$ a los términos tercero y último.

$$\overline{A}+\overline{A}\overline{C}+\overline{A}\overline{B}+\overline{B}\overline{C}$$

Paso 5. Aplicar la regla 10, $\overline{A}+\overline{A}\overline{C}=\overline{A}(1+\overline{C})=\overline{A}$, a los términos primero y segundo.

$$\overline{A}+\overline{A}\overline{B}+\overline{B}\overline{C}$$

Paso 6. Aplicar la regla 10 $[\overline{A}+\overline{A}\overline{B}=\overline{A}(1+\overline{B})=\overline{A}]$ a los términos primero y segundo.

$$\overline{A}+\overline{B}\overline{C}$$

Problema relacionado Simplificar la expresión booleana $\overline{AB}+\overline{AC}+\overline{A}\overline{B}\overline{C}$.

REVISIÓN DE LA SECCIÓN 4.5

1. Simplificar, si es posible, las siguientes expresiones booleanas:
   (a) $A+AB+A\overline{B}C$ **(b)** $(\overline{A}+B)C+ABC$ **(c)** $A\overline{B}C(BD+CDE)+A\overline{C}$
2. Implementar con las puertas lógicas apropiadas cada expresión de la cuestión anterior. Después, implementar la expresión simplificada y comparar el número de puertas empleado en cada caso.

4.6 FORMAS ESTÁNDAR DE LAS EXPRESIONES BOOLEANAS

Todas las expresiones booleanas, independientemente de su forma, pueden convertirse en cualquiera de las dos formas estándar: suma de productos o producto de sumas. La estandarización posibilita que la evaluación, simplificación e implementación de las expresiones booleanas sea mucho más sistemática y sencilla.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Identificar una expresión en forma de suma de productos. ■ Determinar el dominio de una expresión booleana. ■ Convertir cualquier suma de productos a su forma estándar. ■ Evaluar una expresión en forma de suma de productos según los valores binarios. ■ Identificar una expresión en forma de producto de sumas. ■ Convertir cualquier producto de sumas a su forma estándar. ■ Evaluar una expresión en forma de producto de sumas según los valores binarios. ■ Convertir expresiones de una a otra forma estándar.

Suma de productos

▲ *Una suma de productos puede implementarse con una puerta OR y dos o más puertas AND.*

En la Sección 4.1, se ha definido el término producto como un término que es el producto (multiplicación booleana) de literales (variables o sus complementos). Cuando dos o más productos se suman mediante la adición booleana, la expresión resultante se denomina ***suma de productos*** (SOP, *Sum Of Products*). Algunos ejemplos son:

$$AB+ABC$$
$$ABC+CDE+\overline{B}C\overline{D}$$
$$\overline{A}B+\overline{A}B\overline{C}+AC$$

Una suma de productos puede contener también términos de una única variable como en $A+\overline{A}\overline{B}C+BC\overline{D}$. Si volvemos a los ejemplos de simplificación de la sección anterior, puede observarse que cada término de la

expresión resultante era o un producto aislado o una suma de productos. En una expresión con formato de suma de productos, una barra no puede extenderse sobre más de una variable; sin embargo, más de una variable puede tener una barra encima. Por ejemplo, una suma de productos puede contener el término $\bar{A}\bar{B}\bar{C}$ pero no el término $\overline{ABC}$.

Dominio de una expresión booleana. El **dominio** de una expresión booleana es el conjunto de variables contenido en la expresión bien en su forma complementada o no complementada. Por ejemplo, el dominio de la expresión $\bar{A}B + A\bar{B}C$ es el conjunto de variables A, B, C y el dominio de la expresión $AB\bar{C} + C\bar{D}E + \bar{B}C\bar{D}$ es el conjunto de variables A, B, C, D, E.

Implementación AND/OR de una suma de productos. La implementación de una suma de productos simplemente requiere aplicar la operación OR a las salidas de dos o más puertas AND. Una operación AND da lugar a un producto, y la adición de dos o más productos se realiza mediante puertas OR. Por tanto, una expresión suma de productos puede implementarse mediante un circuito lógico AND-OR en el que las salidas de las puertas AND, cuyo número es igual al de productos que contenga la expresión, son las entradas de una puerta OR, como se muestra en la Figura 4.18 para la expresión $AB + BCD + AC$. La salida X de la puerta OR es igual a la suma de productos.

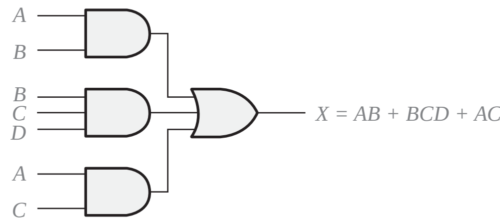


FIGURA 4.18 Implementación de la suma de productos $AB + BCD + AC$.

Implementación NAND/NAND de una suma de productos. Se pueden emplear puertas NAND para implementar una expresión suma de productos. Utilizando sólo puertas NAND se puede obtener una función AND/OR, como se ilustra en la Figura 4.19. El primer nivel de puertas NAND alimenta las entradas de una puerta NAND que actúa como una puerta negativa-OR. Las inversiones de la puerta NAND y las puertas negativa-OR se cancelan y dan como resultado un circuito AND/OR.

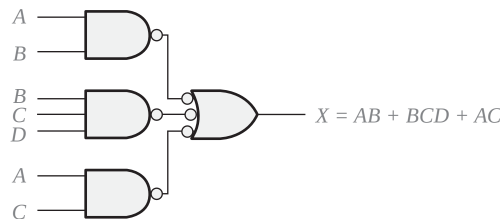


FIGURA 4.19 Esta implementación NAND/NAND es equivalente a la implementación AND/OR de la Figura 4.18.

Conversión de una expresión general a formato suma de productos

Cualquier expresión lógica puede ser transformada a una expresión suma de productos aplicando el álgebra de Boole. Por ejemplo, la expresión $A(B+CD)$ puede convertirse en una suma de productos aplicando la ley distributiva:

$$A\,(B + CD) = AB + ACD$$

EJEMPLO 4.12

Convertir cada una de las siguientes expersiones booleanas a su forma suma de productos:

(a) $AB + B(CD + EF)$ **(b)** $(A + B)(B + C + D)$ **(c)** $\overline{(\overline{A+B})+C}$

Solución

(a) $AB + B(CD + EF) = AB + BCD + BEF$

(b) $(A + B)(B + C + D) = AB + AC + AD + BB + BC + BD$

(c) $\overline{(\overline{A+B})+C} = (\overline{\overline{A+B}})\overline{C} = (A+B)\overline{C} = A\overline{C} + B\overline{C}$

Problema relacionado Convertir $\overline{A}B\overline{C} + (A+\overline{B})(B+\overline{C}+A\overline{B})$ a la forma suma de productos.

Forma estándar de la suma de productos

Hasta ahora, hemos estado viendo sumas de productos en las que algunos de los términos no contenían todas las variables del dominio de la expresión. Por ejemplo, la expresión $\overline{A}B\overline{C} + A\overline{B}D + \overline{A}\overline{B}\overline{C}D$ tiene un dominio formado por las variables A, B, C y D. Sin embargo, el conjunto completo de variables del dominio no está representado en los dos primeros términos de la expresión; es decir, faltan D o $\overline{D}$ en el primer término y C o $\overline{C}$ en el segundo.

Una *suma de productos estándar* es aquella en la que *todas* las variables del dominio aparecen en cada uno de los términos de la expresión. Por ejemplo, $A\overline{B}CD + \overline{A}\overline{B}C\overline{D} + AB\overline{C}\overline{D}$ es una expresión suma de productos estándar. La expresión suma de productos estándar es importante en la construcción de tablas de verdad, lo que se estudiará en la Sección 4.7 y en el método de simplificación de los mapas de Karnaugh, que se aborda en la Sección 4.8. Cualquier expresión suma de productos no estándar (que denominaremos simplemente suma de productos) puede convertirse al formato estándar utilizando el álgebra de Boole.

Conversión de una suma de productos a su forma estándar. Cada término producto de una suma de productos que no contenga todas las variables del dominio puede ampliase a su forma estándar de manera que incluya todas las variables del dominio y sus complementos. Como se muestra en los siguientes pasos, una suma de productos no estándar se convierte a su forma estándar utilizando la regla 6 $(A + \overline{A} = 1)$ de la Tabla 4.1: la suma de una variable y su complemento es igual a 1.

Paso 1. Multiplicar cada término producto no estándar por un término formado por la suma de la variable que falta y su complemento. Con esto se obtienen dos términos producto. Como se sabe, se puede multiplicar por 1 cualquier expresión sin que se altere su valor.

Paso 2. Repetir el paso 1 hasta que todos los términos de la expresión contengan todas las variables o sus complementos del dominio. Al convertir cada producto a su forma estándar, el número de términos producto se duplica por cada variable que falta, como muestra el Ejemplo 4.13.

EJEMPLO 4.13

Convertir la siguiente expresión booleana al formato suma de productos estándar:

$$A\overline{B}C + \overline{A}\overline{B} + AB\overline{C}D$$

Solución El dominio de esta suma de productos es A, B, C, D. Considerando cada término por separado, se comprueba que al primer término, $A\overline{B}C$, le falta la variable D o $\overline{D}$, por lo que multiplicamos dicho término por $D + \overline{D}$ como sigue:

$$A\overline{B}C = A\overline{B}C(D+\overline{D}) = A\overline{B}CD + A\overline{B}C\overline{D}$$

En este caso se obtienen dos productos estándar.

En el segundo término $\overline{A}\overline{B}$ faltan las variables C o $\overline{C}$ y D o $\overline{D}$, por lo que lo multiplicamos por $C+\overline{C}$

$$\overline{A}\overline{B} = \overline{A}\overline{B}(C+\overline{C}) = \overline{A}\overline{B}C + \overline{A}\overline{B}\overline{C}$$

Los dos términos que hemos obtenido carecen de la variable D o $\overline{D}$, por lo que multiplicamos ambos términos por $D+\overline{D}$

$$\begin{aligned}\overline{A}\overline{B} &= \overline{A}\overline{B}C + \overline{A}\overline{B}\overline{C} = \overline{A}\overline{B}C(D+\overline{D}) + \overline{A}\overline{B}\overline{C}(D+\overline{D}) \\ &= \overline{A}\overline{B}CD + \overline{A}\overline{B}C\overline{D} + \overline{A}\overline{B}\overline{C}D + \overline{A}\overline{B}\overline{C}\overline{D}\end{aligned}$$

En este caso, el resultado con cuatro productos estándar.

El tercer término, $AB\overline{C}D$, ya está en forma estándar. La suma de productos estándar completa que obtenemos finalmente es:

$$A\overline{B}C + \overline{A}\overline{B} + AB\overline{C}D = A\overline{B}CD + A\overline{B}C\overline{D} + \overline{A}\overline{B}CD + \overline{A}\overline{B}C\overline{D} + \overline{A}\overline{B}\overline{C}D + \overline{A}\overline{B}\overline{C}\overline{D} + AB\overline{C}D$$

Problema relacionado Convertir la expresión $W\overline{X}Y + \overline{X}Y\overline{Z} + WX\overline{Y}$ a su forma de suma de productos estándar.

Representación binaria de un término producto estándar. Un término producto estándar es igual a 1 sólo para una combinación de los valores de las variables. Por ejemplo, el término producto $A\overline{B}C\overline{D}$ es igual a 1 cuando A = 1, B =0, C =1, D = 0, como se muestra a continuación y es igual a 0 para todas las restantes combinaciones de valores de las variables.

$$A\overline{B}C\overline{D} = 1\cdot\overline{0}\cdot1\cdot\overline{0} = 1\cdot1\cdot1\cdot1 = 1$$

En este caso, el término producto tiene un valor binario de 1010 (diez en decimal).

Recuerde que un término producto se implementa mediante una puerta AND cuya salida es 1 si y sólo si cada una de sus entradas está a 1. Para generar el complemento de las variables cuando es necesario se utilizan inversores.

Una expresión suma de productos es igual a 1 si y sólo si uno o más de los términos productos que forman la expresión es igual a 1.

EJEMPLO 4.14

Determinar los valores binarios para los que la siguiente suma de productos estándar sea igual a 1:

$$ABCD + A\overline{B}\overline{C}D + \overline{A}\overline{B}\overline{C}\overline{D}$$

Solución El término $ABCD$ es igual a 1 cuando A = 1, B = 1, C = 1 y D =1.

$$ABCD = 1\cdot1\cdot1\cdot1 = 1$$

El término $A\overline{B}\overline{C}D$ es igual a 1 cuando A = 1, B = 0, C = 0 y D =1.

$$A\overline{B}\overline{C}D = 1\cdot\overline{0}\cdot\overline{0}\cdot 1 = 1\cdot 1\cdot 1\cdot 1 = 1$$

El término $\overline{A}\overline{B}\overline{C}\overline{D}$ es igual a 1 cuando $A = 0$, $B = 0$, $C = 0$ y $D = 0$.

$$\overline{A}\overline{B}\overline{C}\overline{D} = \overline{0}\cdot\overline{0}\cdot\overline{0}\cdot\overline{0} = 1\cdot 1\cdot 1\cdot 1 = 1$$

La suma de productos es igual a 1 sólo cuando cualquiera de los tres términos o todos son igual a 1.

Problema relacionado Determinar los valores binarios para los que la siguiente expresión suma de productos es igual a 1:

$$\overline{X}YZ + X\overline{Y}Z + XY\overline{Z} + \overline{X}Y\overline{Z} + XYZ$$

¿Es una suma de productos estándar?

Producto de sumas

En la Sección 4.1 se ha definido el término suma como un término formado por la suma (adición booleana) de literales (variables o sus complementos). Cuando dos o más términos suma se multiplican, la expresión resultante es un ***producto de sumas*** (POS, *Product Of Sums*). Algunos ejemplos son:

$$(\overline{A} + B)(A + \overline{B} + C)$$

$$(\overline{A} + \overline{B} + \overline{C})(C + \overline{D} + E)(\overline{B} + C + D)$$

$$(A + B)(A + \overline{B} + C)(\overline{A} + C)$$

Un producto de sumas puede contener términos con una única variable como en $\overline{A}(A + \overline{B} + C)(\overline{B} + \overline{C} + D)$. En una expresión producto de sumas, una barra no puede extenderse nunca sobre más de una variable, aunque más de una variable puede tener una barra encima. Por ejemplo, un producto de sumas puede contener el término $\overline{A} + \overline{B} + \overline{C}$ pero no el $\overline{A + B + C}$.

Implementación de un producto de sumas. La implementación de un producto de sumas requiere simplemente la aplicación de la operación AND a las salidas de dos o más puertas OR. Un sumando se origina mediante la operación OR y el producto de varios términos suma se realiza por medio de la operación AND. Por tanto, un producto de sumas puede implementarse a partir de puertas lógicas OR (cuyo número será igual al de sumandos de la expresión) cuyas salidas se conectan a las entradas de una puerta AND, como muestra la Figura 4.20 para la expresión $(A + B)(B + C + D)(A + C)$. La salida X de la puerta AND es igual al producto de sumas.

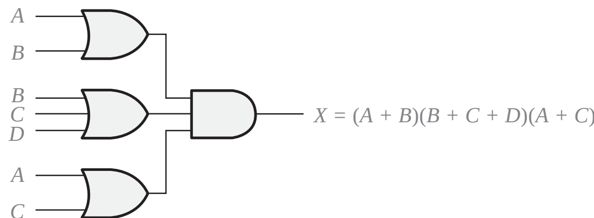


FIGURA 4.20 Implementación del producto de sumas $(A + B)(B + C + D)(A + C)$.

Forma estándar del producto de sumas

Hasta ahora, se han tratado expresiones producto de sumas en las que algunos de los términos no contenían todas las variables del dominio de la expresión. Por ejemplo, la expresión:

$$(A+\overline{B}+C)(A+B+\overline{D})(A+\overline{B}+\overline{C}+D)$$

tiene un dominio formado por las variables A, B, C y D. Observe que el conjunto completo de variables del dominio no está representado en los dos primeros términos de la expresión; es decir, faltan D o $\overline{D}$ en el primer término y C o $\overline{C}$ en el segundo término.

Un producto de sumas estándar es aquel en el que *todas* las variables del dominio o sus complementos aparecen en cada uno de los términos de la expresión. Por ejemplo,

$$(\overline{A}+\overline{B}+\overline{C}+\overline{D})(A+\overline{B}+C+D)(A+B+\overline{C}+D)$$

es un producto de sumas estándar. Cualquier producto de sumas no estándar (que denominaremos simplemente producto de sumas) puede convertirse a su forma estándar mediante el álgebra de Boole.

Conversión de un producto de sumas a su forma estándar. Cada término suma de una expresión producto de sumas que no contenga todas las variables del dominio puede extenderse para obtener su formato estándar incluyendo todas las variables del dominio y sus complementos. Como se establece en los pasos siguientes, un producto de sumas no estándar se convierte a su formato estándar utilizando la regla booleana número 8 $(A\cdot\overline{A}=0)$ de la Tabla 4.1 que establece que una variable multiplicada por su complemento es igual a 0.

Paso 1. Añadir a cada término suma no estándar un término formado por la variable que falta y su complemento. Esto da lugar a la aparición de dos términos suma. Como ya sabemos, se puede sumar 0 a cualquier cosa sin que se altere su valor.

Paso 2. Aplicar la regla 12 de la Tabla 4.1: $A+BC=(A+B)(A+C)$.

Paso 3. Repetir el paso 1 hasta que todos los términos suma resultantes contengan todas las variables del dominio en su forma complementada o no complementada.

EJEMPLO 4.15

Convertir la siguiente expresión booleana a formato producto de sumas:

$$(A+\overline{B}+C)(\overline{B}+C+\overline{D})(A+\overline{B}+\overline{C}+D)$$

Solución

El dominio de este producto de sumas es A, B, C, D. Vamos a considerar término por término. En el primero $A+\overline{B}+C$, falta la variable D o $\overline{D}$, por lo que añadimos $D\overline{D}$ y aplicamos la regla 12 del siguiente modo:

$$A+\overline{B}+C=A+\overline{B}+C+D\overline{D}=(A+\overline{B}+C+D)(A+\overline{B}+C+\overline{D})$$

En el segundo término, $\overline{B}+C+\overline{D}$ falta la variable A o $\overline{A}$**,** por lo que añadimos $A\overline{A}$ y aplicamos la regla 12 como sigue:

$$\overline{B}+C+\overline{D}=\overline{B}+C+\overline{D}+A\overline{A}=(A+\overline{B}+C+\overline{D})(\overline{A}+\overline{B}+C+\overline{D})$$

El tercer término, $A+\overline{B}+\overline{C}+D$, ya está en formato estándar. El producto de sumas estándar de la expresión original es:

$$(A+\overline{B}+C)(\overline{B}+C+\overline{D})(A+\overline{B}+\overline{C}+D)=$$

$$(A+\overline{B}+C+D)(A+\overline{B}+C+\overline{D})(A+\overline{B}+C+\overline{D})(\overline{A}+\overline{B}+C+\overline{D})(A+\overline{B}+\overline{C}+D)$$

Problema relacionado Convertir la expresión $(A+\overline{B})(B+C)$ a su forma producto de sumas estándar.

Representación binaria de un término suma estándar. Un término suma estándar es igual a 0 sólo para una combinación de los valores de las variables. Por ejemplo, el término suma $A+\overline{B}+C+\overline{D}$ es igual a 1 cuando $A = 0$, $B = 1$, $C = 0$ y $D = 1$, como se muestra a continuación y es igual a 1 para todas las restantes combinaciones de valores de las variables.

$$A+\overline{B}+C+\overline{D}=0+\overline{1}+0+\overline{1}=0+0+0+0=0$$

En este caso, el término suma tiene un valor binario de 0101 (cinco en decimal). Recuerde que un término suma se implementa mediante una puerta OR cuya salida es 0 sólo si cada una de sus entradas está a 0. Para generar el complemento de las variables cuando es necesario se utilizan inversores.

Una expresión producto de sumas es igual a 0 si y sólo si uno o más de los términos suma que forman la expresión es igual a 0.

EJEMPLO 4.16

Determinar los valores binarios de las variables para los que la expresión producto de sumas estándar siguiente es igual a 0:

$$(A+B+C+D)(A+\overline{B}+\overline{C}+D)(\overline{A}+\overline{B}+\overline{C}+\overline{D})$$

Solución El término $A + B + C + D$ es igual a 0 cuando $A = 0$, $B = 0$, $C = 0$ y $D = 0$.

$$A + B + C + D = 0 + 0 + 0 + 0 = 0$$

El término $A+\overline{B}+\overline{C}+D$ es igual a 0 cuando $A = 0$, $B = 1$, $C = 1$ y $D = 0$:

$$A+\overline{B}+\overline{C}+D=0+\overline{1}+\overline{1}+0=0+0+0+0=0$$

El término $\overline{A}+\overline{B}+\overline{C}+\overline{D}$ es igual a 0 cuando $A = 1$, $B = 1$, $C = 1$ y $D = 1$.

$$\overline{A}+\overline{B}+\overline{C}+\overline{D}=\overline{1}+\overline{1}+\overline{1}+\overline{1}=0+0+0+0=0$$

La expresión producto de sumas es igual a 0 cuando cualquiera de los tres términos suma es igual a 0.

Problema relacionado Determinar los valores binarios para los que la siguiente expresión producto de sumas es igual a 0:

$$(X+\overline{Y}+Z)(\overline{X}+Y+Z)(X+Y+\overline{Z})(\overline{X}+\overline{Y}+\overline{Z})(X+\overline{Y}+\overline{Z})$$

¿Es un producto de sumas estándar?

Conversión de una suma de productos estándar en un producto de sumas estándar

Los valores binarios de los términos producto en una suma de productos estándar dada no aparecen en su producto de sumas estándar equivalente. Asimismo, los valores binarios que no están representados en una suma de productos sí aparecen en el producto de sumas equivalente. Por tanto, para pasar de la suma de productos estándar al producto de sumas estándar hay que realizar los siguientes pasos:

Paso 1. Evaluar cada término producto de la expresión suma de productos. Es decir, determinar los números binarios que representan estos términos.

Paso 2. Determinar todos los números binarios no incluidos al realizar la evaluación del paso 1.

Paso 3. Escribir los términos suma equivalente para cada valor binario del paso 2 y expresarlos en forma producto de sumas.

Utilizando un procedimiento similar, se puede pasar de un producto de sumas a una suma de productos.

EJEMPLO 4.17

Convertir la siguiente suma de productos en su expresión equivalente como producto de sumas:

$$\overline{A}\overline{B}\overline{C}+\overline{A}B\overline{C}+\overline{A}BC+A\overline{B}C+ABC$$

Solución El resultado de la evaluación es la siguiente

$$000 + 010 + 011 + 101 + 111$$

Puesto que son tres las variables que conforman el dominio de esta expresión, existe un total de ocho (2^3) posibles combinaciones. La suma de productos contiene cinco de estas combinaciones, luego la expresión producto de sumas debe contener las otras tres que son 001, 100 y 110. Recuerde que estos son los valores binarios que hacen que cada término suma sea igual a cero. La expresión producto de sumas equivalente es la siguiente:

$$(A+B+\overline{C})(\overline{A}+B+C)(\overline{A}+\overline{B}+C)$$

Problema relacionado Sustituyendo los valores binarios en cada término, verificar que las expresiones suma de productos y producto de sumas de este ejemplo son equivalentes.

REVISIÓN DE LA SECCIÓN 4.6

1. Determinar si cada una de las expresiones siguientes es una suma de productos o un producto de sumas. Indicar si se trata de una forma estándar.
   (a) $AB+\overline{A}BD+\overline{A}C\overline{D}$
   (b) $(A+\overline{B}+C)(A+B+\overline{C})$
   (c) $\overline{A}BC+AB\overline{C}$
   (d) $A(A+\overline{C})(A+B)$
2. Convertir las sumas de productos de la cuestión 1 a la forma estándar.
3. Convertir los productos de sumas de la cuestión 1 a la forma estándar

4.7 EXPRESIONES BOOLEANAS Y TABLAS DE VERDAD

Todas las expresiones booleanas pueden convertirse fácilmente en tablas de verdad utilizando los valores binarios de cada término de la expresión. La tabla de verdad es una forma muy común, en un formato muy conciso, de expresar el funcionamiento lógico de un circuito. Además, las expresiones suma de productos y producto de sumas pueden determinarse mediante las tablas de verdad. Las tablas de verdad pueden encontrarse en las hojas de especificaciones y en otros textos relativos al funcionamiento de los circuitos y sistemas digitales.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Pasar una expresión suma de productos estándar a su tabla de verdad. ■ Pasar un producto de sumas estándar a su tabla de verdad. ■ Obtener una expresión estándar a partir de su tabla de verdad. ■ Interpretar correctamente los datos de una tabla de verdad.

Conversión de una suma de productos a tabla de verdad

Como se ha establecido en la Sección 4.6, una suma de productos es igual a 1 sólo si y sólo si al menos uno de los productos es igual a 1. Una tabla de verdad es sencillamente la lista de las posibles combinaciones de valores de las variables de entrada y sus correspondientes valores de salida (1 o 0). Para una expresión cuyo dominio es de dos variables, existen cuatro combinaciones distintas de estas variables ($2^2 = 4$). Para una expresión cuyo dominio tiene tres variables, existen ocho ($2^3 = 8$) combinaciones posibles de dichas variables. Para una expresión con un dominio de cuatro variables, existen dieciséis combinaciones diferentes de dichas variables ($2^4 = 16$), etc.

El primer paso para construir una tabla de verdad consiste en enumerar todas las posibles combinaciones de los valores de las variables de la expresión. A continuación, hay que pasar la suma de productos a su formato estándar, si no lo está ya. Por último, se escribe un 1 en la columna de salida (X) para cada valor binario que hace que la suma de productos estándar sea 1, y se escribe un 0 para los restantes valores. Este procedimiento se ilustra en el Ejemplo 4.18.

EJEMPLO 4.18

Desarrollar una tabla de verdad para la expresión suma de productos estándar $\overline{A}\overline{B}C + A\overline{B}\overline{C} + ABC$.

Solución Existen tres variables en el dominio, por lo que hay ocho posibles combinaciones de valores binarios de las variables, como se muestra en las tres columnas

	Entradas		**Salida**	
A	***B***	***C***	***X***	**Término producto**
0	0	0	0	
0	0	1	1	$\overline{A}\overline{B}C$
0	1	0	0	
0	1	1	0	
1	0	0	1	$A\overline{B}\overline{C}$
1	0	1	0	
1	1	0	0	
1	1	1	1	ABC

TABLA 4.6

de la izquierda de la Tabla 4.6. Los valores binarios que hacen que los términos producto de la expresión sean igual a 1 son $\overline{A}\overline{B}C : 001$; $A\overline{B}\overline{C} : 100$ y ABC: 111. Para cada uno de estos valores binarios, se escribe un 1 en la columna de salida, como se indica en la tabla. Para cada una de las restantes combinaciones, se escribe un 0 en la columna de salida.

Problema relacionado Crear una tabla de verdad para la expresión suma de productos estándar $\overline{A}B\overline{C} + A\overline{B}C$.

Conversión de un producto de sumas a tabla de verdad

Recuerde que un producto de sumas es igual a 0 sólo si y sólo si al menos uno de los términos suma es igual a 0. Para construir la tabla de verdad de un producto de sumas, basta con enumerar todas las posibles combinaciones de valores binarios de las variables del mismo modo que se hace para una suma de productos. A continuación, hay que pasar el producto de sumas a su formato estándar, si no lo está ya. Por último, se escribe un 0 en la columna de salida (X) para cada valor binario que hace que la suma de productos estándar sea 0, y se escribe un 1 para los restantes valores binarios. Este procedimiento se ilustra en el Ejemplo 4.19.

EJEMPLO 4.19

Desarrollar una tabla de verdad para la expresión producto de sumas estándar siguiente:

$$(A+B+C)(A+\overline{B}+C)(A+\overline{B}+\overline{C})(\overline{A}+B+\overline{C})(\overline{A}+\overline{B}+C)$$

Solución Existen tres variables en el dominio, por lo que hay ocho posibles combinaciones de valores binarios de las variables, como se muestra en las tres columnas de la izquierda de la Tabla 4.7. Los valores binarios que hacen que los términos suma de la expresión sean igual a 0 son $A + B + C$: 000; $A+\overline{B}+C : 010$; $A+\overline{B}+\overline{C} : 011$; $\overline{A}+B+\overline{C} : 101$ y $\overline{A}+\overline{B}+C : 110$. .Para cada uno de estos valores binarios, se escribe un 0 en la columna de salida, como se indica en la tabla. Para cada una de las restantes combinaciones, se escribe un 1 en la columna de salida.

Entradas			Salida	
A	*B*	*C*	*X*	Término suma
0	0	0	0	$(A+B+C)$
0	0	1	1	
0	1	0	0	$(A+\overline{B}+C)$
0	1	1	0	$(A+\overline{B}+\overline{C})$
1	0	0	1	
1	0	1	0	$(\overline{A}+B+\overline{C})$
1	1	0	0	$(\overline{A}+\overline{B}+C)$
1	1	1	1	

TABLA 4.7

Observe que la tabla de verdad de este ejemplo es la misma que la del Ejemplo 4.18. Esto significa que la suma de productos del ejemplo anterior y el producto de sumas de este ejemplo son equivalentes.

Problema relacionado Crear una tabla de verdad para la expresión producto de sumas estándar: $(A+\overline{B}+C)(A+B+\overline{C})(\overline{A}+\overline{B}+\overline{C})$

Determinación de las expresiones estándar a partir de una tabla de verdad

Para determinar la expresión de la suma de productos estándar representada por una tabla de verdad se enumeran todos los valores de las variables de entrada para los que la salida es 1. Cada valor binario se convierte en el correspondientes término producto, reemplazando cada 1 por la variable y cada 0 por la variable complementada. Por ejemplo, el valor binario 1010 se transforma en un término producto de la manera siguiente:

$$1010 \rightarrow A\overline{B}C\overline{D}$$

Si sustituimos podemos comprobar que el término producto es 1:

$$A\overline{B}C\overline{D} = 1 \cdot \overline{0} \cdot 1 \cdot \overline{0} = 1 \cdot 1 \cdot 1 \cdot 1 = 1$$

Para determinar el producto de sumas estándar representado por una tabla de verdad se enumeran todos los valores binarios para los que la salida es 0. A continuación, se convierte cada valor binario en el correspondiente término suma, reemplazando cada 1 por la variable complementada y cada 0 por la variable. Por ejemplo, el número binario 1001 se pasa a término suma de la manera siguiente:

$$1001 \rightarrow \overline{A}+B+C+\overline{D}$$

Si sustituimos podemos comprobar que el término suma es 0:

$$\overline{A}+B+C+\overline{D} = \overline{1}+0+0+\overline{1} = 0+0+0+0 = 0$$

EJEMPLO 4.20

A partir de la tabla de verdad de la Tabla 4.8, determinar la expresión suma de productos estándar y la expresión producto de sumas estándar equivalente.

Entradas			**Salida**
A	***B***	***C***	***X***
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

TABLA 4.8

Solución En la columna de salida hay cuatro 1s y los correspondientes valores binarios son 011, 100, 110 y 111. Convertir estos valores binarios a términos producto como sigue:

$$011 \rightarrow \overline{A}BC$$
$$100 \rightarrow A\overline{B}\overline{C}$$
$$110 \rightarrow AB\overline{C}$$
$$111 \rightarrow ABC$$

La expresión suma de productos estándar resultante para la salida X es:

$$X = \overline{A}BC + A\overline{B}\overline{C} + AB\overline{C} + ABC$$

Para el producto de sumas, la salida es 0 para los valores binarios 000, 001, 010 y 101. Estos valores binarios se convierten en términos suma como sigue:

$$000 \rightarrow A + B + C$$
$$001 \rightarrow A + B + \overline{C}$$
$$010 \rightarrow A + \overline{B} + C$$
$$101 \rightarrow A + B + \overline{C}$$

La expresión producto de sumas estándar resultantes para la salida X es:

$$X = (A + B + C)(A + B + \overline{C})(A + \overline{B} + C)(A + B + \overline{C})$$

Problema relacionado Sustituyendo los valores binarios, demostrar que las expresiones suma de productos y productos de sumas obtenidas en este ejemplo son equivalentes; es decir, para cualquier número binario que se elija ambas deben ser 1 o 0, dependiendo del valor binario.

REVISIÓN DE LA SECCIÓN 4.7

1. Si una determinada expresión booleana tiene un dominio de cinco variables, ¿cuántos valores binarios tendrá su tabla de verdad?
2. En una determinada tabla de verdad, la salida es 1 para el valor binario 0110. Convertir este valor binario en el correspondiente término producto usando las variables W, X, Y y Z.
3. En una determinada tabla de verdad, la salida es 0 para el valor binario 1100. convertir este valor binario en el correspondiente término suma usando las variables W, X, Y y Z.

4.8 MAPAS DE KARNAUGH

Un mapa de Karnaugh proporciona un método sistemático de simplificación de expresiones booleanas y, si se aplica adecuadamente, genera las expresiones suma de productos y producto de sumas más simples posibles, conocidas como expresiones mínimas. Como hemos visto, la efectividad de la simplificación algebraica depende de nuestra familiaridad con las leyes, reglas y teoremas del álgebra de Boole y de nuestra habilidad para aplicarlas. Por otro lado, el mapa de Karnaugh es básicamente una "receta" para la simplificación.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Construir un mapa de Karnaugh de tres o cuatro variables. ■ Determinar el valor binario de cada celda en un mapa de Karnaugh. ■ Determinar el término producto estándar representado en cada celda de un mapa de Karnaugh. ■ Explicar la adyacencia de celdas e identificar celdas adyacentes.

▲ *El propósito de un mapa de Karnaugh es simplificar una expresión booleana.*

Un ***mapa de Karnaugh*** es similar a una tabla de verdad, ya que muestra todos los valores posibles de las variables de entrada y la salida resultante para cada valor. En lugar de organizar en filas y columnas como una tabla de verdad, el mapa de Karnaugh es una matriz de **celdas** en la que cada celda representa un valor binario de las variables de entrada. Las celdas se organizan de manera que la simplificación de una determinada expresión consiste en agrupar adecuadamente las celdas. Los mapas de Karnaugh se pueden utilizar para expresiones de dos, tres, cuatro y cinco variables, pero nos ocuparemos únicamente de los casos de tres y cuatro variables para ilustrar los principios. La Sección 4.11 aborda el caso de cinco variables utilizando un mapa de Karnaugh de 32 celdas. Existe otro método, que queda fuera del propósito de este libro, denominado método de Quine-McClusky, que puede emplearse para un número mayor de variables.

El número de celdas de un mapa de Karnaugh es igual al número total de posibles combinaciones de las variables de entrada, al igual que el número de filas de una tabla de verdad. Para tres variables, el número de celdas necesarias es de $2^3 = 8$. Para cuatro variables, el número de celdas es de $2^4 = 16$.

Mapa de Karnaugh de tres variables

El mapa de Karnaugh de tres variables es una matriz de ocho celdas, como se muestra en la Figura 4.21(a). En este caso, *A*, *B* y *C* se emplean para denominar a las variables, aunque podían haberse usado cualesquiera otras letras. Los valores binarios de *A* y *B* se encuentran en el lado izquierdo (observe la secuencia) y los valores de *C* se colocan en la parte superior. El valor de una determinada celda es el valor binario de *A* y *B*, en la parte izquierda de la misma fila combinado con el valor de *C* en la parte superior de la misma columna. Por ejemplo, la celda de la esquina superior izquierda tiene un valor binario de 000 y la celda inferior derecha tiene un valor binario de 101. La Figura 4.21(b) muestra los términos producto estándar representados por cada celda del mapa de Karnaugh.

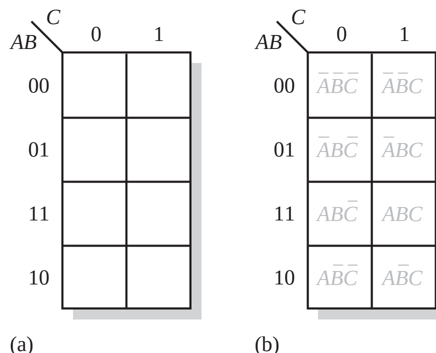


FIGURA 4.21 Mapa de Karnaugh de tres variables que muestra los términos producto.

Mapa de Karnaugh de cuatro variables

El mapa de Karnaugh de cuatro variables es una matriz de dieciséis celdas, como se muestra en la Figura 4.22(a). Los valores binarios de *A* y *B* se encuentran en el lado izquierdo y los valores de *C* y *D* se colocan

en la parte superior. El valor de una determinada celda es el valor binario de *A* y *B*, en la parte izquierda de la misma fila combinado con los valores binarios de *C* y *D* en la parte superior de la misma columna. Por ejemplo, la celda de la esquina superior derecha tiene un valor binario de 0010 y la celda inferior derecha tiene un valor binario de 1010. En la Figura 4.22(b) se indican los términos producto estándar representados por cada celda del mapa de Karnaugh de cuatro variables.

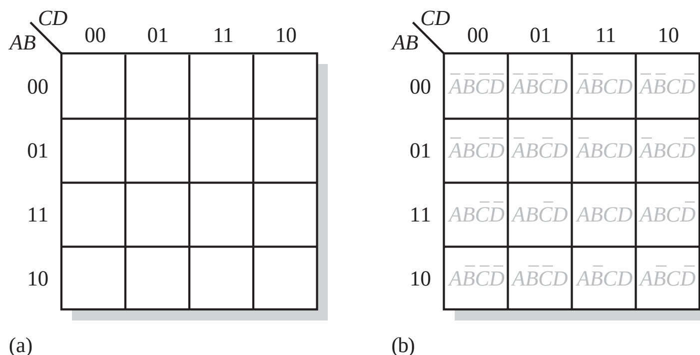


FIGURA 4.22 Mapa de Karnaugh de cuatro variables.

Adyacencia de celdas

▲ *Las celdas que sólo difieren en una variable son adyacentes.*

▲ *Las celdas con valores que difieren en más de una variable no son adyacentes.*

Las celdas de un mapa de Karnaugh se disponen de manera que sólo cambia una única variable entre celdas adyacentes. La **adyacencia** se define por un cambio de una única variable. Las celdas que difieren en una única variable son adyacentes. Por ejemplo, en el mapa de tres variables, la celda 010 es adyacente a las celdas 000, 011 y 110. La celda 010 no es adyacente a la celda 001, ni a la celda 111, ni a la celda 100 ni a la celda 101.

Físicamente, cada celda es adyacente a las celdas que están situadas inmediatas a ella por cualquiera de sus cuatro lados. Un celda no es adyacente a aquellas celdas que tocan diagonalmente alguna de sus esquinas. Además, las celdas de la fila superior son adyacentes a las de la fila inferior y las celdas de la columna izquierda son adyacentes a las situadas en la columna de la derecha. Esto se denomina adyacencia

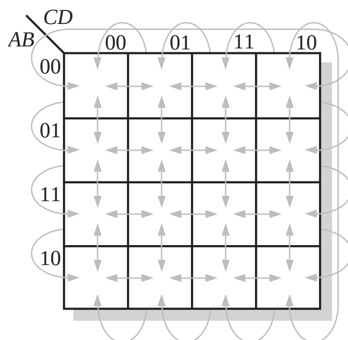


FIGURA 4.23 Celdas adyacentes en un mapa de Karnaugh son aquellas que sólo difieren en una variable. Las flechas apuntan a la celdas adyacentes.

cíclica, ya que podemos pensar que el mapa de Karnaugh se dobla de forma que se toquen los extremos superior e inferior como si fuera un cilindro o los extremos de la derecha e izquierda para formar la misma figura. La Figura 4.23 ilustra la adyacencia de celdas en un mapa de cuatro variables, aunque se aplican las mismas reglas de adyacencia a los mapas de Karnaugh con cualquier número de celdas.

REVISIÓN DE LA SECCIÓN 4.8

1. En un mapa de Karnaugh de 3 variables, ¿cuál es el valor binario de cada una de las siguientes celdas?:
   (a) esquina superior izquierda **(b)** esquina inferior derecha
   (c) esquina inferior izquierda **(d)** esquina superior derecha
2. ¿Cuál es el término producto estándar de cada celda de la cuestión 1 para las variables X, Y y Z?
3. Repetir la cuestión 1 para un mapa de 4 variables.
4. Repetir la cuestión 2 para un mapa de 4 variables utilizando las variables W, X, Y y Z.

4.9 MINIMIZACIÓN DE UNA SUMA DE PRODUCTOS MEDIANTE EL MAPA DE KARNAUGH

Como se ha establecido en la sección anterior, el mapa de Karnaugh se utiliza para reducir expresiones booleanas a su expresión mínima. Una expresión suma de productos minimizada está formada por el mínimo número de términos producto posibles con el mínimo número de variables por término. Generalmente, una expresión suma de productos minimizada puede implementarse mediante un número de puertas menor que su expresión estándar, lo cual constituye la finalidad del proceso de simplificación.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Representar una expresión suma de productos en un mapa de Karnaugh. ■ Combinar los unos del mapa en grupos máximos. ■ Determinar el término producto mínimo para cada grupo del mapa. ■ Combinar los términos producto mínimo para formar una expresión suma de productos mínima. ■ Convertir una tabla de verdad en un mapa de Karnaugh para simplificar la expresión representada. ■ Utilizar las condiciones “indiferentes” en un mapa de Karnaugh.

Mapa de Karnaugh de una suma de productos estándar

Por cada término de la expresión suma de productos, se coloca un 1 en el mapa de Kanaugh en la celda correspondiente al valor del producto. Se coloca un 1 en la celda correspondiente al valor de un término producto. Por ejemplo, para el término $A\bar{B}C$, se escribiría un 1 en la celda 101 de un mapa de Karnaugh de tres variables.

Cuando una expresión suma de productos se ha reflejado por completo en el mapa de Karnaugh, en dicho mapa habrá tantos 1s como términos producto tenga la suma de productos estándar. Las celdas que no contienen un 1 son aquellas para las que la expresión es igual a 0. Normalmente, cuando se trabaja con una expresión suma de productos, los 0s no se incluyen en el mapa. Los siguientes pasos y la Figura 4.24 muestra cómo completar los mapas de Karnaugh.

Paso 1. Determinar el valor binario de cada término producto de la suma de productos estándar. Tras un poco de práctica, podrá realizar la evaluación de términos mentalmente.

Paso 2. A medida que evaluamos cada término, colocamos un 1 en el mapa de Karnaugh en la celda que tiene el mismo valor que dicho término producto.

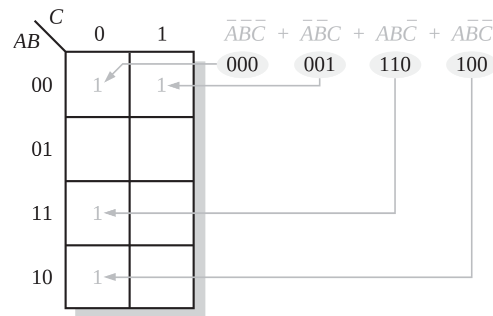


FIGURA 4.24 Ejemplo de transformación a mapa de Karnaugh de una suma de productos estándar.

EJEMPLO 4.21

Transformar la siguiente suma de productos estándar en un mapa de Karnaugh:

$$\overline{A}\overline{B}C + \overline{A}B\overline{C} + AB\overline{C} + ABC$$

Solución La expresión se evalúa como se muestra a continuación. Se escribe un 1 en el mapa de Karnaugh de 3 variables de la Figura 4.24 por cada producto estándar de la expresión.

$$\overline{A}\overline{B}C + \overline{A}B\overline{C} + AB\overline{C} + ABC$$

001 010 110 111

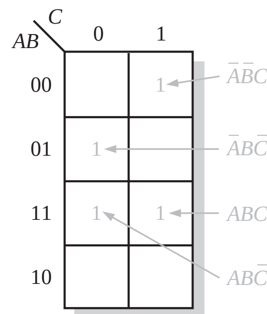


FIGURA 4.25

Problema relacionado Transformar la expresión estándar de la suma de productos $\overline{A}BC + A\overline{B}C + A\overline{B}\overline{C}$ en un mapa de Karnaugh.

EJEMPLO 4.22

Transformar la siguiente suma de productos estándar en un mapa de Karnaugh:

$$\bar{A}\bar{B}CD + \bar{A}B\bar{C}\bar{D} + AB\bar{C}D + ABCD + AB\bar{C}\bar{D} + \bar{A}\bar{B}\bar{C}D + A\bar{B}C\bar{D}$$

Solución La expresión se evalúa como se muestra a continuación. Se coloca un 1 en el mapa de Karnaugh de la Figura 4.26 por cada producto estándar de la expresión.

$$\bar{A}\bar{B}CD + \bar{A}B\bar{C}\bar{D} + AB\bar{C}D + ABCD + AB\bar{C}\bar{D} + \bar{A}\bar{B}\bar{C}D + A\bar{B}C\bar{D}$$

0011 0100 1101 1111 1100 0001 1010

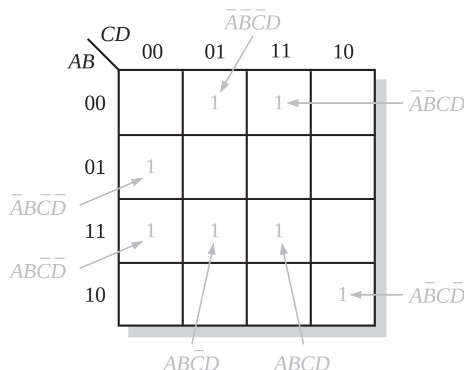


FIGURA 4.26

Problema relacionado Transformar la siguiente expresión estándar suma de productos en un mapa de Karnaugh.

$$\bar{A}BC\bar{D} + ABC\bar{D} + AB\bar{C}\bar{D} + ABCD$$

Mapa de Karnaugh de una suma de productos no estándar

Antes de poder utilizar un mapa de Karnaugh, las expresiones booleanas deben estar en su forma estándar. Si una expresión no lo está, se pasará al formato estándar mediante el procedimiento descrito en la Sección 4.6 o mediante desarrollo numérico. Dado que, en cualquier caso, las expresiones tienen que evaluarse antes de pasarlas al mapa de Karnaugh, el desarrollo numérico es quizá el método más eficaz.

Desarrollo numérico de un producto no estándar. Recuerde que a un término en forma no estándar le faltan una o más variables en su expresión. Por ejemplo, supongamos que uno de los productos de una determinada suma de productos de 3 variables es $A\bar{B}$. Este término se puede desarrollar numéricamente para obtener una expresión estándar de la manera siguiente. En primer lugar, se escribe el valor binario de las dos variables y le añadimos un 0 que corresponde a la variable que falta $\bar{C}$: 100. A continuación, escribimos el valor binario de las dos variables y añadimos un 1 para la variable que falta C: 101. Los dos números binarios resultantes son los valores de los términos de la suma de productos estándar $A\bar{B}\bar{C}$ y $A\bar{B}C$.

Veamos otro ejemplo, supongamos que uno de los términos producto de una expresión de 3 variables es B (recuerde que una variable única se considera como un término producto en una expresión suma de productos). Este término puede expandirse numéricamente a su forma estándar de la siguiente manera: se escribe el valor binario de la variable; a continuación, se añaden todos los posibles valores de las variables que faltan A y C del siguiente modo:

B
010
011
110
111

Los cuatro números binarios resultantes son los valores correspondientes a los términos de la suma de productos estándar $\overline{A}B\overline{C}, \overline{A}BC, AB\overline{C}$ y ABC.

EJEMPLO 4.23

Transformar la siguiente expresión suma de productos en un mapa de Karnaugh: $\overline{A} + A\overline{B} + AB\overline{C}$.

Solución Obviamente, la suma de productos no está en formato estándar, ya que cada término no contiene las tres variables. En el primer término faltan dos variables, en el segundo falta una variable y el tercero sí es un término estándar. En primer lugar, desarrollamos los términos numéricamente de la siguiente manera:

$\overline{A}$	$+A\overline{B}$	$+AB\overline{C}$
000	100	110
001	101	
010		
011		

Cada uno de los valores binarios resultantes se traslada al mapa, colocando un 1 en la celda apropiada del mapa de Karnaugh de 3 variables de la Figura 4.27.

AB \ C	0	1
00	1	1
01	1	1
11	1	
10	1	1

FIGURA 4.27

Problema relacionado Transformar la expresión suma de productos $BC + \overline{A}\overline{C}$ en un mapa de Karnaugh.

EJEMPLO 4.24

Transformar la siguiente expresión suma de productos en un mapa de Karnaugh:

$$\overline{B}\overline{C} + A\overline{B} + AB\overline{C} + A\overline{B}C\overline{D} + \overline{A}\overline{B}\overline{C}D + A\overline{B}CD$$

Solución Obviamente, la suma de productos no está en formato estándar, ya que cada término no contiene las cuatro variables. En los términos primero y segundo faltan dos variables, en el tercer término falta una variable y el resto de los términos sí son estándar. En primer lugar, desarrollamos los términos numéricamente para incluir las variables que faltan de la siguiente manera:

$\overline{B}\overline{C}$	$A\overline{B}$	$+AB\overline{C}$	$+A\overline{B}C\overline{D}$	$+\overline{A}\overline{B}\overline{C}D$	$+A\overline{B}CD$
0000	1000	1100	1010	0001	1011
0001	1001	1101			
1000	1010				
1001	1011				

Cada uno de los valores binarios resultantes se traslada al mapa, colocando un 1 en la celda apropiada del mapa de Karnaugh de 4 variables de la Figura 4.28. Observe que algunos de los valores de la expresión desarrollada son redundantes.

AB \ CD	00	01	11	10
00	1	1		
01				
11	1	1		
10	1	1	1	1

FIGURA 4.28

Problema relacionado Transformar la expresión $A + \overline{C}D + AC\overline{D} + \overline{A}BC\overline{D}$ en un mapa de Karnaugh.

Simplificación de una suma de productos mediante el mapa de Karnaugh

El proceso que genera una expresión que contiene el menor número posible de términos con el mínimo número de variables posibles se denomina ***minimización***. Después de haber obtenido el mapa de Karnaugh de una suma de productos, la expresión suma de productos mínima se obtiene agrupando los 1s y determinando la expresión suma de productos mínima a partir del mapa.

Agrupación de unos. Podemos agrupar los unos del mapa de Karnaugh de acuerdo con las reglas siguientes, rodeando las celdas adyacentes que contengan unos. La finalidad es maximizar el tamaño de los grupos y minimizar el número de estos grupos.

1. Un grupo tiene que contener 1, 2, 4, 8 ó 16 celdas, valores que se corresponden con las potencias de 2. En el caso de un mapa de Karnaugh de 3 variables, el grupo máximo puede contener $2^3 = 8$ celdas.
2. Cada celda de un grupo tiene que ser adyacente a una o más celdas del mismo grupo, pero no todas las celdas del grupo tienen que ser adyacentes entre sí.
3. Incluir siempre en cada grupo el mayor número posible de 1s de acuerdo a la regla número 1.
4. Cada 1 del mapa tiene que estar incluido en al menos un grupo. Los 1s que ya pertenezcan a un grupo pueden estar incluidos en otro, siempre que los grupos que se solapen contengan 1s no comunes.

EJEMPLO 4.25

Agrupar los 1s en cada uno de los mapas de Karnaugh de la Figura 4.29.

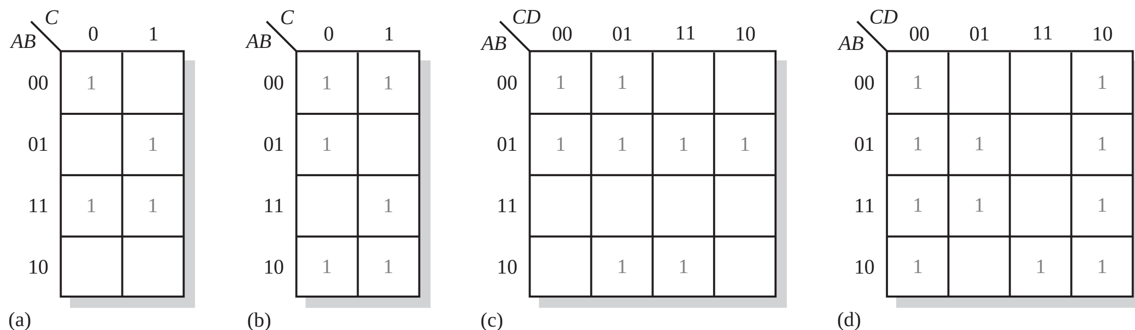


FIGURA 4.29

Solución En la Figura 4.30 se muestran los grupos. En algunos casos, puede existir más de una forma de agrupar los 1s para formar grupos máximos.

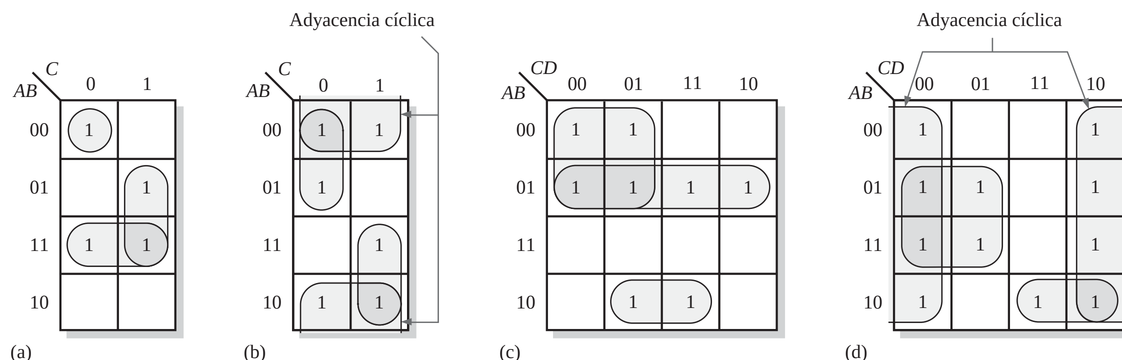


FIGURA 4.30

Problema relacionado Determinar si existen otras formas de agrupar los 1s en la Figura 4.30, para obtener un número mínimo de grupos máximos.

Determinación de la expresión suma de productos mínima a partir del mapa. Cuando todos los 1s que representan los términos productos estándar de una expresión se han trasladado al mapa y se han agrupado adecuadamente, comienza el proceso de obtención de la suma de productos mínima. Para encontrar los términos mínimos y la expresión suma de productos mínima se aplican las siguientes reglas:

1. Agrupar las celdas que contienen 1s. Cada grupo de celdas que contiene 1s da lugar a un término producto compuesto por todas las variables que aparecen en el grupo en sólo una forma (no complementada o complementada). Las variables que aparecen complementadas y sin complementar dentro del mismo grupo se eliminan. A éstas se les denomina *variables contradictorias*.

2. Determinar la operación producto mínima para cada grupo.

(a) Para un mapa de 3 variables:

(1) Un grupo formado por 1 celda da lugar a un término producto de 3 variables.

(2) Un grupo formado por 2 celdas da lugar a un término producto de 2 variables.

(3) Un grupo formado por 4 celdas da lugar a un término de 1 variable.

(4) Un grupo formado por 8 celdas indica que la expresión vale 1.

(b) Para un mapa de 4 variables:

(1) Un grupo formado por 1 celda da lugar a un término producto de 4 variables.

(2) Un grupo formado por 2 celdas da lugar a un término producto de 3 variables.

(3) Un grupo formado por 4 celdas da lugar a un término producto de 2 variables.

(4) Un grupo formado por 8 celdas da lugar a un término de 1 variable.

(5) Un grupo formado por 16 celdas indica que la expresión vale 1.

3. Cuando se han obtenido todos los términos producto mínimos a partir del mapa de Karnaugh, se suman para obtener la expresión suma de productos mínima.

EJEMPLO 4.26

Determinar los productos para el mapa de Karnaugh de la Figura 4.31 y escribir la expresión suma de productos mínima resultante.

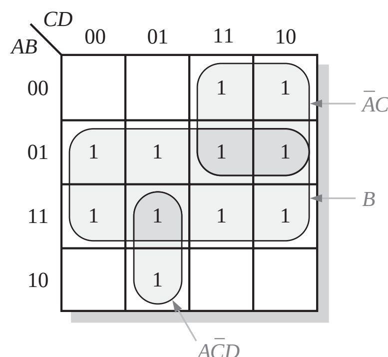


FIGURA 4.31

Solución Se eliminan las variables que aparecen complementadas y no complementadas en un mismo grupo. En la Figura 4.31, el producto para el grupo de 8 celdas es B, ya que las celdas de dicho grupo contienen las variables A y $\bar{A}$, C y $\bar{C}$, y D

y $\overline{D}$, que se eliminan. El grupo de 4 celdas contiene las variables $B, \overline{B}, D$ y $\overline{D}$, quedando las variables $\overline{A}$ y C, que forman el término producto $\overline{A}C$. El grupo de 2 celdas contiene B y $\overline{B}$, quedando las variables A, $\overline{C}$ y D que forman el término producto $A\overline{C}D$. Observe cómo se utiliza el solapamiento para maximizar el tamaño de los grupos. La suma de productos mínima resultantes es la suma de estos términos producto:

$$B + \overline{A}C + A\overline{C}D$$

Problema relacionado En el mapa de Karnaugh de la Figura 4.31, añadir un 1 a la celda inferior derecha (1010) y determinar la expresión suma de productos resultante.

EJEMPLO 4.27

Determinar los productos para cada uno de los mapas de Karnaugh de la Figura 4.32 y escribir las correspondientes expresiones suma de productos mínima resultante.

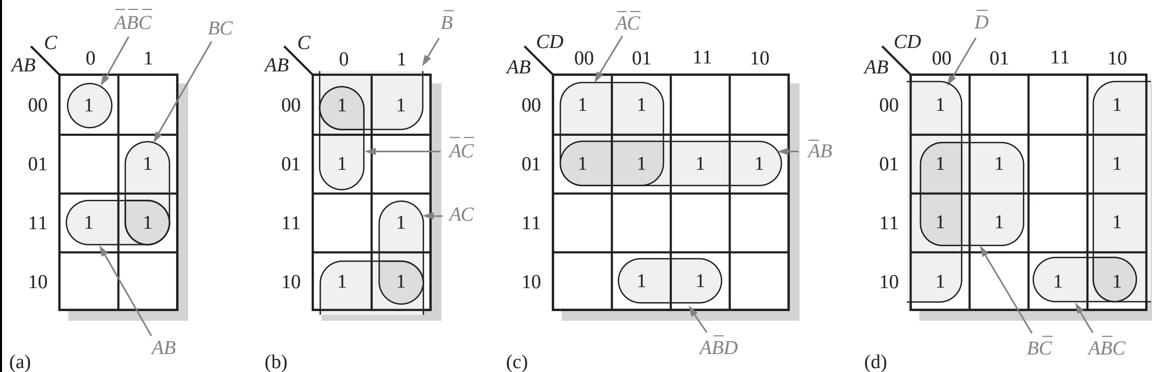


FIGURA 4.32

Solución En la Figura 4.32 se muestran los productos mínimos resultantes para cada grupo. La expresión suma de productos mínima para cada uno de los mapas de Karnaugh de la figura son:

(a) $AB + BC + \overline{A}\overline{B}\overline{C}$ **(b)** $\overline{B} + \overline{A}\overline{C} + AC$

(c) $\overline{A}B + \overline{A}\overline{C} + A\overline{B}D$ **(d)** $\overline{D} + A\overline{B}C + B\overline{C}$

Problema relacionado En el mapa de Karnaugh de la Figura 4.32(d), añadir un 1 a la celda 0111 y determinar la expresión suma de productos resultante.

EJEMPLO 4.28

Utilizar un mapa de Karnaugh para minimizar la siguiente expresión suma de productos estándar:

$$A\overline{B}C + \overline{A}BC + \overline{A}\overline{B}C + \overline{A}\overline{B}\overline{C} + A\overline{B}\overline{C}$$

Solución Los valores binarios de la expresión son:

101 + 011 + 011 + 000 + 100

La suma de productos estándar se pasa al mapa y las celdas se agrupan como se muestra en la Figura 4.33.

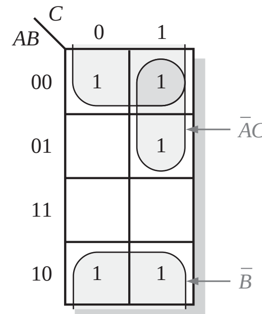


FIGURA 4.33

Observe que el grupo de 4 celdas en los extremos del mapa que incluye las filas superior e inferior de 1s. El 1 restante se incluye en otro grupo superpuesto de dos celdas. El grupo de los cuatro 1s da lugar a un término de una sola variable, $\overline{B}$. Esto se deduce del hecho de que, dentro del grupo, $\overline{B}$ es la única variable que no cambia de celda a celda. El grupo de los dos 1s da lugar al producto de dos variables $\overline{A}C$. Este término se determina observando que, dentro de este grupo, $\overline{A}$ y C no cambian de una celda a la siguiente. Se ha indicado el término producto correspondiente a cada grupo. La expresión suma de productos mínima resultante es:

$$\overline{B} + \overline{A}C$$

Tenga presente que esta expresión mínima es equivalente a la expresión estándar original.

Problema relacionado Utilizando un mapa de Karnaugh, simplificar la siguiente expresión suma de productos estándar:

$$X\overline{Y}Z + XY\overline{Z} + \overline{X}YZ + \overline{X}Y\overline{Z} + X\overline{Y}\overline{Z} + XYZ$$

EJEMPLO 4.29

Utilizar un mapa de Karnaugh para minimizar la siguiente expresión suma de productos:

$$\overline{B}\overline{C}\overline{D} + \overline{A}B\overline{C}\overline{D} + AB\overline{C}\overline{D} + \overline{A}\overline{B}CD + A\overline{B}CD + \overline{A}\overline{B}C\overline{D} + \overline{A}BC\overline{D} + ABC\overline{D} + A\overline{B}C\overline{D}$$

Solución El primer término $\overline{B}\overline{C}\overline{D}$ tiene que desarrollarse en los términos $A\overline{B}\overline{C}\overline{D}$ y $\overline{A}\overline{B}\overline{C}\overline{D}$ para obtener la suma de productos estándar, que a continuación se trasladará a un mapa, donde se agrupan las celdas como se muestra en la Figura 4.34.

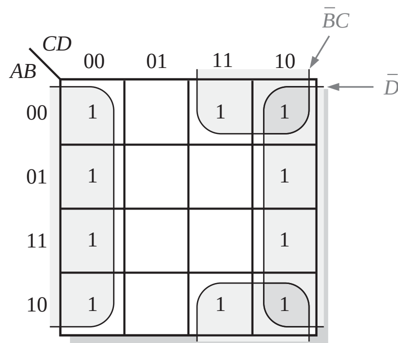


FIGURA 4.34

Observe que ambos grupos tienen adyacencia "cíclica" de celdas. Se puede formar el grupo de ocho celdas, ya que las dos columnas exteriores son adyacentes. El grupo de cuatro celdas se forma tomando los dos restantes 1s, puesto que las celdas superior e inferior son adyacentes. Se indica el término producto para cada grupo y la expresión suma de productos mínima resultante es:

$$\bar{D} + \bar{B}C$$

Tenga presente que esta expresión mínima es equivalente a la expresión estándar original.

Problema relacionado Mediante un mapa de Karnaugh simplifique la siguiente expresión suma de productos:

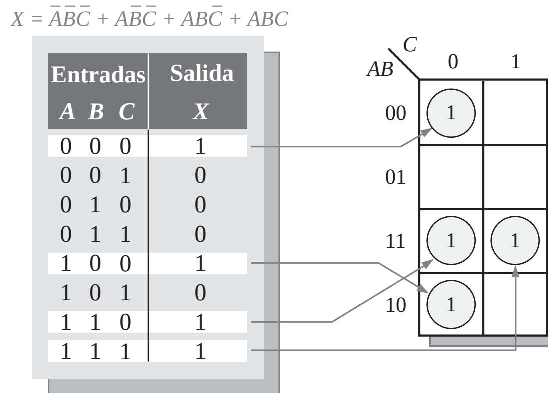
$$\bar{W}\bar{X}\bar{Y}\,\bar{Z} + W\bar{X}YZ + W\bar{X}\bar{Y}Z + \bar{W}YZ + W\bar{X}\bar{Y}\,\bar{Z}$$

Obtención directa del mapa de Karnaugh a partir de la tabla de verdad

Hemos visto cómo las expresiones booleanas se transforman en mapas de Karnaugh. Ahora aprenderá cómo pasar de una tabla de verdad a un mapa de Karnaugh. Recuerde que una tabla de verdad proporciona la salida de una expresión booleana para todas las posibles combinaciones de las variables de entrada. En la Figura 4.35 se facilita un ejemplo de expresión booleana junto con su tabla de verdad. Observe que la salida X es 1 para cuatro distintas combinaciones de las variables de entrada. Los 1s de la columna de salida de la tabla de verdad se trasladan directamente al mapa de Karnaugh, a las celdas correspondientes a los valores asociados de las combinaciones de variables de entrada, como muestra la Figura 4.35. En esta figura puede ver que tanto la expresión booleana, la tabla de verdad como el mapa de Karnaugh son sólo distintas maneras de representar una función lógica.

Condiciones indiferentes

Algunas veces se producen situaciones en las que algunas combinaciones de las variables de entrada no están permitidas. Por ejemplo, recuerde que en el código BCD, visto en el Capítulo 2, existían seis combinaciones no válidas: 1010, 1011, 1100, 1101, 1110 y 1111. Dado que estos estados no permitidos no ocurren nunca en una aplicación que emplee el código BCD, pueden considerarse como términos ***indiferentes*** con respecto a su efecto en la salida. Esto significa que a estos términos se les puede asignar tanto un 1 como un 0 en la salida; realmente no son importantes dado que nunca van a generarse.

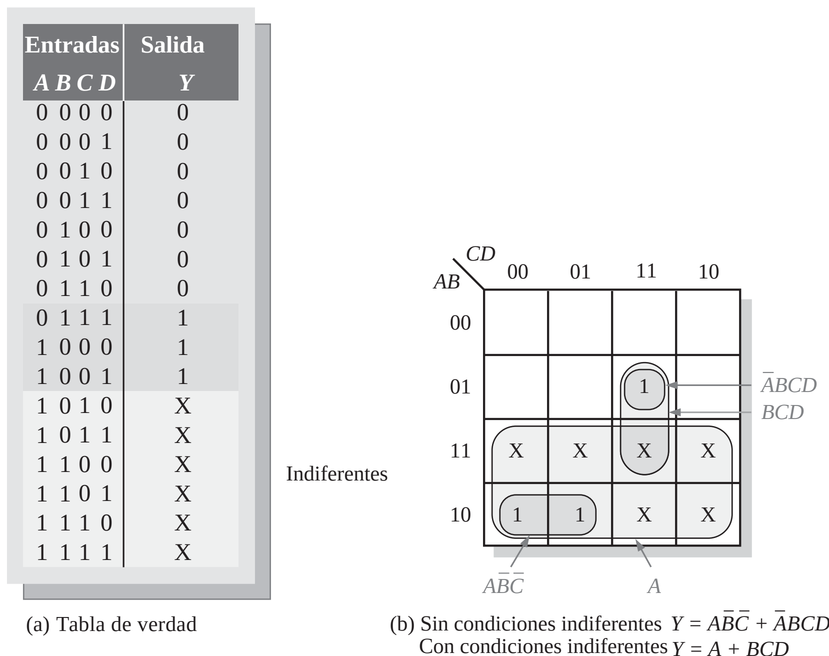

$X = \bar{A}\bar{B}\bar{C} + A\bar{B}\bar{C} + AB\bar{C} + ABC$

Entradas A B C	Salida X
0 0 0	1
0 0 1	0
0 1 0	0
0 1 1	0
1 0 0	1
1 0 1	0
1 1 0	1
1 1 1	1

FIGURA 4.35 Ejemplo de obtención directa de un mapa de Karnaugh a partir de una tabla de verdad.

Los términos "indiferentes" pueden utilizarse para aprovechar mejor el método del mapa de Karnaugh. La Figura 4.36 muestra que, para cada término indiferente, se escribe una X en la celda. Cuando se agrupan los 1s, las X pueden ser consideradas también como 1s para agrandar los grupos, o como 0s si no obtenemos ninguna ventaja. Cuanto mayor sea el grupo, más sencillo será el término resultante.

La tabla de verdad de la Figura 4.36(a) describe una función lógica que tiene sólo la salida igual a 1 cuando el código BCD correspondiente al 7, 8 o 9 está presente en las entradas. Si las condiciones "indiferentes" se emplean como 1s, la expresión resultante para la función es $A + BCD$, como se indica en la parte (b) de la figura. Si las condiciones "indiferentes" no se establecen como 1s, la expresión resultante es $A\bar{B}\bar{C} + \bar{A}BCD$, por lo que puede concluirse que los términos indiferentes pueden aprovecharse para obtener una expresión más sencilla.



Entradas A B C D	Salida Y
0 0 0 0	0
0 0 0 1	0
0 0 1 0	0
0 0 1 1	0
0 1 0 0	0
0 1 0 1	0
0 1 1 0	0
0 1 1 1	1
1 0 0 0	1
1 0 0 1	1
1 0 1 0	X
1 0 1 1	X
1 1 0 0	X
1 1 0 1	X
1 1 1 0	X
1 1 1 1	X

(a) Tabla de verdad

(b) Sin condiciones indiferentes $Y = A\bar{B}\bar{C} + \bar{A}BCD$
Con condiciones indiferentes $Y = A + BCD$

FIGURA 4.36 Ejemplo de la utilización de las condiciones "indiferentes" para simplificar una expresión.

REVISIÓN DE LA SECCIÓN 4.9

1. Explicar los mapas de Karnaugh de 3 y 4 variables.
2. Agrupar los 1s y escribir la expresión suma de productos simplificada para el mapa de Karnaugh de la Figura 4.25.
3. Escribir la expresión estándar de la suma de productos original de cada uno de los mapas de Karnaugh de la Figura 4.32.

4.10 MINIMIZACIÓN DE UN PRODUCTO DE SUMAS MEDIANTE EL MAPA DE KARNAUGH

En la sección anterior estudiamos la minimización de una expresión suma de productos mediante los mapas de Karnaugh. En esta sección, nos vamos a centrar en las expresiones producto de sumas. Los métodos son muy similares, excepto que ahora se trata de productos de sumas, en los que los 0s representan los términos suma estándar y se colocan en el mapa de Karnaugh en lugar de los 1s.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Transformar un producto de sumas estándar en un mapa de Karnaugh. ■ Combinar los 0s del mapa para formar grupos máximos. ■ Determinar el término suma mínimo para cada grupo del mapa. ■ Combinar los términos suma mínimos para formar el producto de sumas mínimo. ■ Utilizar el mapa de Karnaugh para convertir productos de sumas en sumas de productos.

Conversión de una expresión producto de sumas estándar a mapa de Karnaugh

Para un producto de sumas en forma estándar, se introduce un 0 en el mapa de Karnaugh por cada término suma de la expresión. Cada 0 se sitúa en la celda correspondiente al valor de un término suma. Por ejemplo, para la suma $A+\overline{B}+C$, se escribe un 0 en la celda 010 del mapa de Karnaugh de 3 variables.

Cuando un producto de sumas se ha trasladado por completo al mapa, habrá tantos 0s en el mapa de Karnaugh, como términos suma en la expresión del producto de sumas estándar. Las celdas que no contienen un 0 son aquellas para las que la expresión vale 1. Generalmente, cuando se trabaja con productos de sumas, los 1s no se escriben. Los siguientes pasos junto con la Figura 4.37 ilustran este proceso.

Paso 1. Determinar el valor binario de cada término suma del producto de sumas estándar. Éste es el valor binario que hace que dicho término sea igual a 0.

Paso 2. Cada vez que se evalúa un término suma, se introduce un 0 en la correspondiente celda del mapa de Karnaugh.

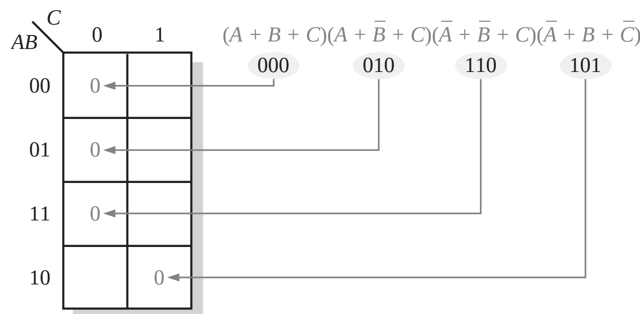


FIGURA 4.37 Ejemplo de obtención del mapa de Karnaugh de un producto de sumas estándar.

EJEMPLO 4.30

Transformar la siguiente expresión suma de productos estándar en un mapa de Karnaugh:

$$(\overline{A}+\overline{B}+C+D)(\overline{A}+B+\overline{C}+\overline{D})(A+B+\overline{C}+D)(\overline{A}+\overline{B}+\overline{C}+\overline{D})(A+B+\overline{C}+\overline{D})$$

Solución La expresión se evalúa como se indica a continuación y se coloca un 0 en el mapa de Karnaugh de 4 variables de la Figura 4.38 por cada término suma estándar de la expresión.

$$\underset{1100}{(\overline{A}+\overline{B}+C+D)}\underset{1011}{(\overline{A}+B+\overline{C}+\overline{D})}\underset{0010}{(A+B+\overline{C}+D)}\underset{1111}{(\overline{A}+\overline{B}+\overline{C}+\overline{D})}\underset{0011}{(A+B+\overline{C}+\overline{D})}$$

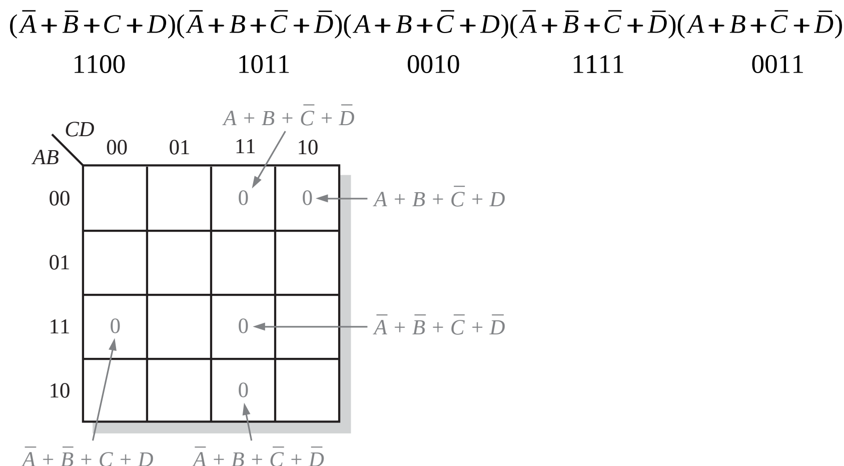


FIGURA 4.38

Problema relacionado Transformar la siguiente expresión suma de productos estándar en un mapa de Karnaugh.

$$(A+\overline{B}+\overline{C}+D)(A+B+C+\overline{D})(A+B+C+D)(\overline{A}+B+\overline{C}+D)$$

Simplificación mediante el mapa de Karnaugh de expresiones producto de sumas

El proceso de minimización de un producto de sumas es básicamente el mismo que para una expresión suma de productos, excepto que ahora hay que agrupar los ceros para generar el mínimo número de términos suma, en lugar de los 1s para obtener el número mínimo de términos producto. Las reglas para agrupar los 0s son las mismas que para agrupar los 1s, y son las que se han estudiado en la Sección 4.9.

EJEMPLO 4.31

Utilizar un mapa de Karnaugh para minimizar la siguiente expresión producto de sumas estándar:

$$(A+B+C)(A+B+\overline{C})(A+\overline{B}+C)(A+\overline{B}+\overline{C})(\overline{A}+\overline{B}+C)$$

Deducir también la expresión suma de productos equivalente.

Solución Las combinaciones de valores binarios de la expresión son

(0 + 0 + 0) (0 + 0 + 1) (0 + 1 + 0) (0 + 1 + 1) (1 + 1 + 0)

La expresión de la suma de productos estándar se traslada al mapa de Karnaugh y las celdas se agrupan como se muestra en la Figura 4.39

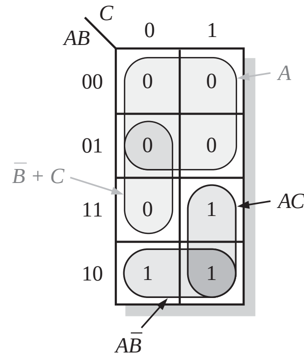


FIGURA 4.39

Observe que el 0 de la celda 110 se incluye en un grupo de dos celdas, utilizando el 0 del grupo de cuatro celdas. El término suma para cada grupo se muestra en la figura y la expresión suma de productos mínima resultante es:

$$A(\overline{B}+C)$$

Tenga en cuenta que esta expresión suma de productos mínima es equivalente a la expresión suma de productos estándar.
Agrupando los 1s como se indica en la áreas de color gris se obtiene una expresión suma de productos que es equivalente a agrupar los ceros.

$$AC + A\overline{B} = A(\overline{B}+C)$$

Problema relacionado Utilizar un mapa de Karnaugh para simplificar la siguiente suma de productos estándar:

$$(X+\overline{Y}+Z)(X+\overline{Y}+\overline{Z})(\overline{X}+\overline{Y}+Z)(\overline{X}+Y+Z)$$

EJEMPLO 4.32

Utilizar un mapa de Karnaugh para minimizar la siguiente expresión producto de sumas:

$$(B+C+D)(A+B+\overline{C}+D)(\overline{A}+B+C+\overline{D})(A+\overline{B}+C+D)(\overline{A}+\overline{B}+C+D)$$

Solución El primer término tiene que desarrollarse en los términos $\overline{A}+B+C+D$ y $A + B + C + D$ para obtener una expresión producto de sumas estándar, que luego debe pasarse a un mapa de Karnaugh, y agrupar las celdas como se muestra en la Figura 4.40. El término suma correspondiente a cada grupo se indica en la figura y la expresión producto de sumas mínima resultante es:

$$(C+D)(A+B+D)(\overline{A}+B+C)$$

Tenga en cuenta que este producto de sumas mínimo es equivalente al producto de sumas estándar original.

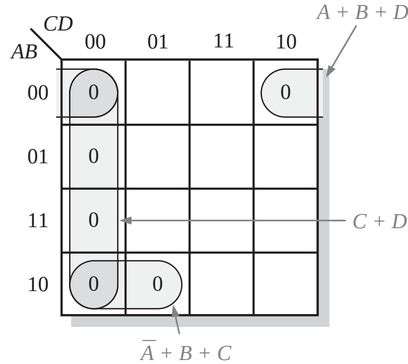


FIGURA 4.40

Problema relacionado Utilizar un mapa de Karnaugh para simplificar la siguiente expresión producto de sumas:

$$(W+\overline{X}+Y+\overline{Z})(W+X+Y+Z)(W+\overline{X}+\overline{Y}+Z)(\overline{W}+\overline{X}+Z)$$

Conversión entre suma de productos y productos de sumas mediante el mapa de Karnaugh

Cuando un producto de sumas se traslada a un mapa de Karnaugh, puede fácilmente pasarse a la suma de productos equivalente directamente a partir de dicho mapa. También, dado un mapa de Karnaugh de una suma de productos, el producto de sumas equivalente puede obtenerse directamente a partir del mapa. Esto proporciona una excelente manera de comparar ambas formas mínimas de una expresión, para determinar si una de ellas puede implementarse con menos puertas que la otra.

Para un producto de sumas, todas las celdas que no contienen 0s contienen 1s, de lo que se deriva su expresión suma de productos. De igual manera, para una suma de productos, todas las celdas que no contienen 1s contendrán 0s, de los que se obtiene la expresión producto de sumas. El Ejemplo 4.33 ilustra esta conversión.

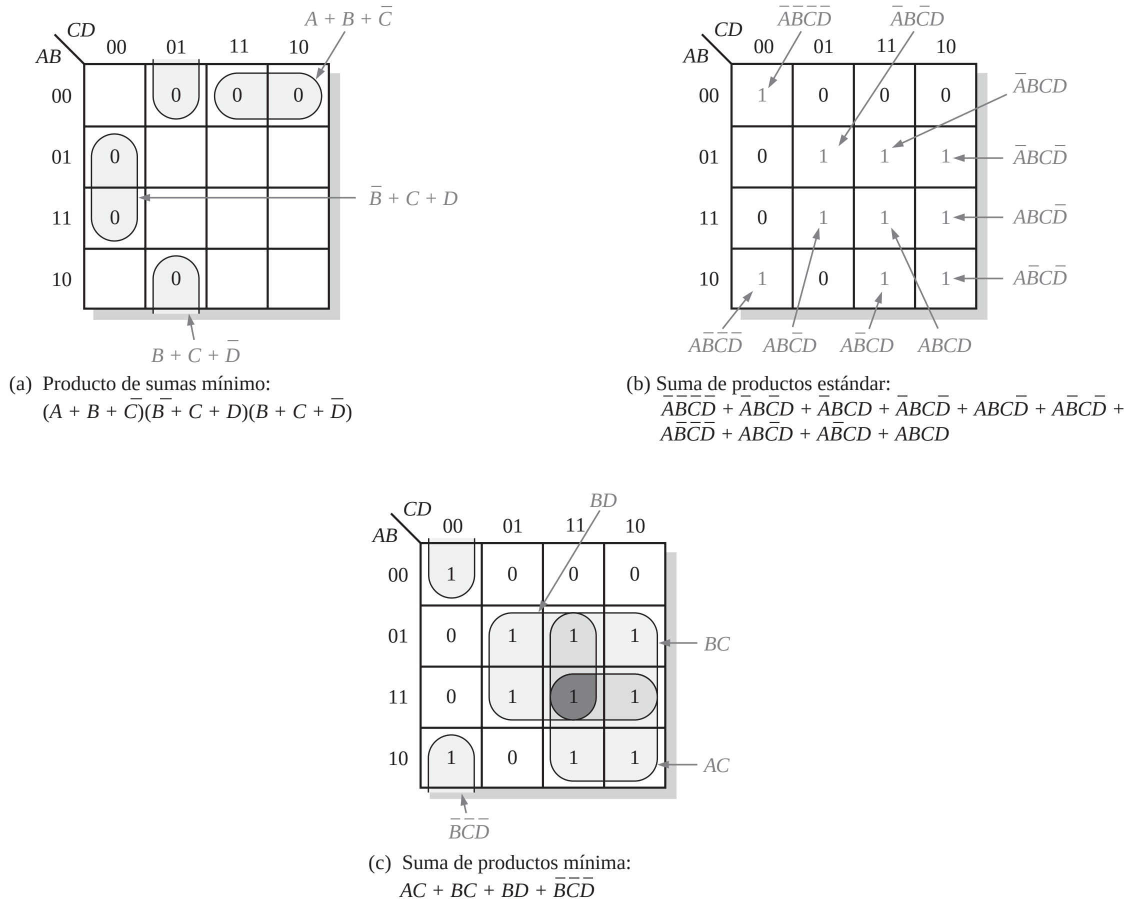
EJEMPLO 4.33

Utilizando un mapa de Karnaugh, convertir el siguiente producto de sumas estándar en un producto de sumas mínimo, una suma de productos estándar y una suma de productos mínima.

$$(\overline{A}+\overline{B}+C+D)(A+\overline{B}+C+D)(A+B+C+\overline{D})$$
$$(A+B+\overline{C}+\overline{D})(\overline{A}+B+C+\overline{D})(A+B+\overline{C}+D)$$

Solución Los ceros de la expresión producto de sumas estándar se transforman y agrupan para obtener el producto de sumas mínimo, como se indica en la Figura 4.41(a). En la Figura 4.41(b), se añaden 1s en las celdas que no contienen 0s.

De cada celda que contenga un 1, se obtiene un término producto estándar, como se indica. Estos términos producto forman la expresión suma de productos estándar. En la Figura 4.41(c), se agrupan los 1s y se obtiene una expresión suma de productos mínima.



(a) Producto de sumas mínimo: $(A + B + \overline{C})(\overline{B} + C + D)(B + C + \overline{D})$

(b) Suma de productos estándar: $\overline{A}\,\overline{B}\,\overline{C}\,\overline{D} + \overline{A}B\overline{C}D + \overline{A}BCD + \overline{A}BC\overline{D} + ABC\overline{D} + A\overline{B}C\overline{D} + A\overline{B}\,\overline{C}\,\overline{D} + AB\overline{C}D + A\overline{B}CD + ABCD$

(c) Suma de productos mínima: $AC + BC + BD + \overline{B}\,\overline{C}\,\overline{D}$

FIGURA 4.41

Problema relacionado Utilizar un mapa de Karnaugh para convertir la siguiente expresión a su forma suma de productos mínima:

$$(W + \overline{X} + Y + \overline{Z})(\overline{W} + X + \overline{Y} + \overline{Z})(\overline{W} + \overline{X} + \overline{Y} + Z)(\overline{W} + \overline{X} + \overline{Z})$$

REVISIÓN DE LA SECCIÓN 4.10

1. ¿Cuál es la diferencia entre pasar a mapa de Karnaugh un producto de sumas y una suma de productos?
2. ¿Cuál es el término suma estándar expresado con las variables *A*, *B*, *C* y *D* para un 0 en la celda 1011 del mapa de Karnaugh?
3. ¿Cuál es el término producto estándar expresado con las variables *A*, *B*, *C* y *D* para un 1 en la celda 0010 del mapa de Karnaugh?

4.11 MAPA DE KARNAUGH DE CINCO VARIABLES

Las funciones booleanas de cinco variables pueden simplificarse mediante un mapa de Karnaugh de 32 celdas. Realmente, para construir un mapa de 5 variables se utilizan dos mapas de cuatro variables (con 16 celdas cada uno). Ya conocemos la adyacencia de celdas en los mapas de 4 variables y cómo se forman los grupos de celdas que contengan 1s para simplificar una suma de productos. Luego todo lo que se necesita aprender para manejar cinco variables es la adyacencia de celdas entre los dos mapas de 4 variables y cómo agruparlas.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Determinar la adyacencia de celdas en un mapa de 5 variables. ■ Formar grupos que contengan el máximo número de celdas en un mapa de 5 variables. ■ Minimizar las expresiones booleanas de 5 variables utilizando el mapa de Karnaugh.

Un mapa de Karnaugh de cinco variables (*ABCDE*) puede crearse utilizando dos mapas de 4 variables, con los que ya estamos familiarizados. Cada mapa contiene 16 celdas con todas las posibles combinaciones de las variables *B, C, D* y *E*. Un mapa es para *A*=0, mientras que el otro es para *A*=1, como se muestra en la Figura 4.42.

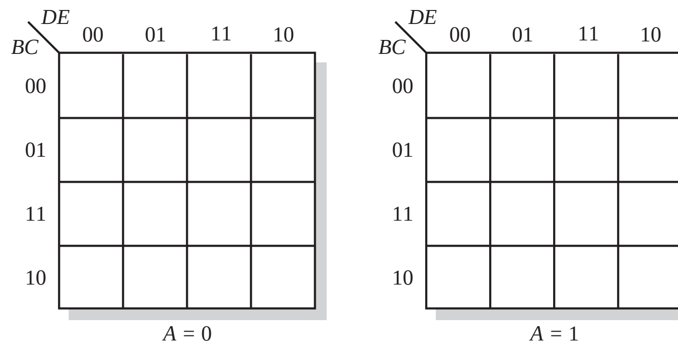


FIGURA 4.42 Mapa de Karnaugh de 5 variables.

Adyacencia de celdas

Ya sabemos cómo determinar celdas adyacentes dentro de un mapa de cuatro variables. La mejor manera de visualizar la adyacencia de celdas entre los dos mapas de 16 celdas consiste en imaginar que el mapa $A = 0$ está colocado encima del mapa $A = 1$. Cada celda del mapa $A = 0$ es adyacente con la celda que está justo debajo en el mapa $A = 1$.

Para ilustrar esto, la Figura 4.43 muestra un ejemplo con cuatro grupos, con los mapas en disposición tridimensional. Los 1s de las celdas en gris más claro forman un grupo de 8 bits (cuatro correspondientes al mapa $A = 0$ combinadas con cuatro del mapa $A = 1$). Los 1s de las celdas marcadas con un degradado de grises forman un grupo de 4 bits. Los 1s de las celdas de la esquina inferior izquierda constituyen un grupo de 4 bits sólo en el mapa $A = 0$. El 1 de la celda gris oscuro del mapa $A = 1$ se agrupa con el 1 de la celda gris más claro de la parte inferior derecha del mapa $A = 0$ para formar un grupo de 2 bits.

Determinación de la expresión booleana. La expresión booleana suma de productos original que está dibujada en el mapa de Karnaugh de la Figura 4.43 contiene diecisiete términos de cinco variables, ya que existen dieci-

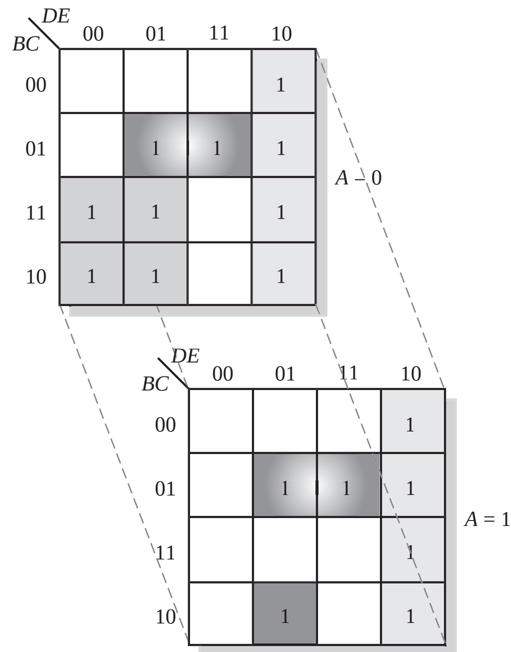


FIGURA 4.43 Ilustración de la agrupación de 1s en celdas adyacentes de un mapa de 5 variables.

siete 1s en el mapa. Como ya sabemos, sólo las variables que no cambian de no complementada a complementada dentro de un grupo permanecen en la expresión correspondiente a ese grupo. La expresión simplificada se obtiene a partir del mapa de la manera siguiente:

- El término para el grupo de ocho 1s marcado en gris claro es $D\overline{E}$.
- El término para el grupo de cuatro 1s marcado en gris degradado es $\overline{B}CE$.
- El término para el grupo de cuatro 1s de la esquina inferior izquierda del mapa $A = 0$ es $\overline{A}B\overline{D}$.
- El término para la celda gris más oscuro agrupada con la celda en gris más claro es $B\overline{C}\overline{D}E$.

Combinando estos términos en la expresión suma de productos simplificada tenemos

$$X = D\overline{E} + \overline{B}CE + \overline{A}B\overline{D} + B\overline{C}\overline{D}E$$

EJEMPLO 4.34

Utilizar un mapa de Karnaugh para minimizar la siguiente expresión suma de productos de 5 variables:

$$X = \overline{A}\overline{B}\overline{C}\overline{D}\overline{E} + \overline{A}\overline{B}C\overline{D}\overline{E} + \overline{A}BC\overline{D}\overline{E} + \overline{A}B\overline{C}\overline{D}\overline{E} + \overline{A}\overline{B}\overline{C}\overline{D}E + \overline{A}BC\overline{D}E$$
$$+ \overline{A}BCDE + A\overline{B}\overline{C}\overline{D}\overline{E} + A\overline{B}\overline{C}\overline{D}E + ABC\overline{D}E + ABCDE + A\overline{B}CDE$$

Solución En la Figura 4.44, se traslada la suma de productos al mapa de Karnaugh y se realizan las agrupaciones indicando los términos correspondientes. Combinando estos términos se obtiene la siguiente expresión suma de productos minimizada:

$$X + \overline{A}\overline{D}\overline{E} + \overline{B}\overline{C}\overline{D} + BCE + ABCE$$

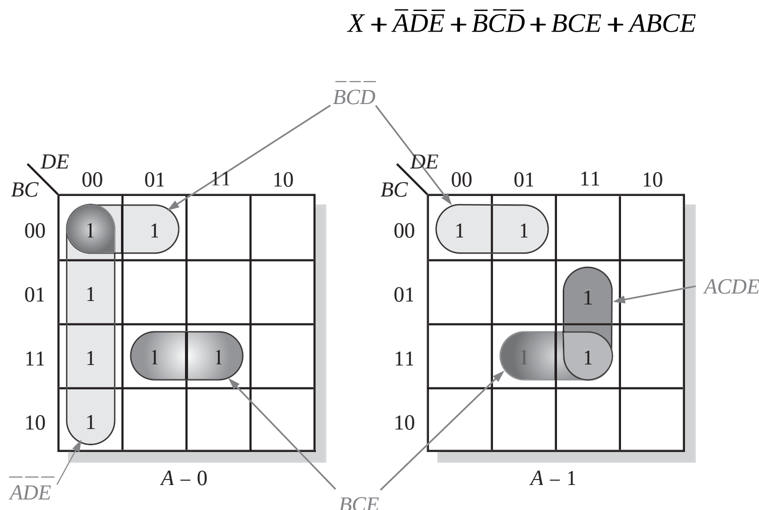


FIGURA 4.44

Problema relacionado Minimizar la siguiente expresión:

$$X = \overline{A}\overline{B}\overline{C}\overline{D}\overline{E} + \overline{A}\overline{B}C\overline{D}\overline{E} + \overline{A}BC\overline{D}\overline{E} + \overline{A}B\overline{C}\overline{D}\overline{E} + A\overline{B}\overline{C}\overline{D}\overline{E} + A\overline{B}C\overline{D}E + ABC\overline{D}\overline{E} + AB\overline{C}\overline{D}\overline{E}$$
$$+ \overline{A}\overline{B}\overline{C}D\overline{E} + \overline{A}\overline{B}CD\overline{E} + \overline{A}BCD\overline{E} + \overline{A}B\overline{C}D\overline{E} + A\overline{B}\overline{C}D\overline{E} + A\overline{B}CD\overline{E} + ABCD\overline{E} + AB\overline{C}D\overline{E}$$

REVISIÓN DE LA SECCIÓN 4.11

1. ¿Por qué un mapa de Karnaugh de 5 variables requiere 32 celdas?
2. ¿Cuál es la expresión representada por un mapa de Karnaugh de cinco variables en el que cada celda contiene un 1?

4.12 VHDL (OPCIONAL)

Esta sección opcional proporciona una breve introducción al lenguaje VHDL y su finalidad no es la de enseñar la estructura y la sintaxis completas del lenguaje. Para obtener información más detallada, consulte las referencias incluidas en la nota a pie de página. Los lenguajes de descripción hardware (HDL, hardware description language) son herramientas que permiten introducir los diseños lógicos, utilizando texto, que se emplean para implementar circuitos lógicos en dispositivos lógicos programables. Aunque el VHDL proporciona múltiples métodos para describir un circuito lógico, aquí sólo vamos a ver los ejemplos de programación más sencillos y directos de introducción del diseño mediante texto.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Establecer los elementos fundamentales del VHDL. ■ Escribir un programa VHDL simple.

La V de VHDL* viene de VHSIC (*Very High Speed Integrated Circuit*) y, claro está, HDL es el acrónimo de *Hardware Description Language*. Como ya hemos mencionado, ***VHDL*** es un lenguaje estándar adoptado por

---

*** *Véase* Floyd, Thomas. 2003. *Digital Fundamentals with VHDL.* Prentice Hall; Pellerin, David y Taylor, Douglas. 1997. *VHDL Made Easy!* Prentice Hall; Bhasker, Jayaram. 1999. *A VHDL Primer,* 3 ed. Prentice Hall.**

el IEEE (*Institute of Electrical and Electronics Engineers*) y se designa como IEEE Std. 1076-1993. VHDL es un lenguaje complejo y exhaustivo y utilizarlo para sacarle el máximo partido posible exige un gran esfuerzo y tener experiencia.

VHDL proporciona tres métodos básicos para describir un circuito digital por software: *comportamental*, *flujo de datos* y *estructural.* Esta exposición vamos a restringirla al método de flujo de datos en el que se escriben instrucciones de tipo booleano para describir un circuito lógico. Tenga en cuenta que el VHDL, asi como otros lenguajes HDL, es una herramienta que permite implementar diseños digitales y es, por tanto, un medio para conseguir un fin y no un fin en sí mismo.

Es relativamente fácil escribir programas para describir circuitos lógicos simples en VHDL. Los operadores lógicos son las siguientes palabras clave VHDL: **and**, **or, not, nand, nor, xor** y **xnor**. Los dos elementos fundamentales en cualquier programa VHDL son la entidad y la arquitectura y deben utilizarse juntos. La **entidad** describe una determinada función lógica en función de sus entradas externas y sus salidas, denominadas puertos. La **arquitectura** describe la operación interna de la función lógica.

En su forma más simple, el elemento entidad (**entity**) consta de tres instrucciones. La primera de ellas asigna un nombre a una función lógica; la segunda instrucción, denominada instrucción *port* y que se indenta, especifica las entradas y las salidas; y la tercera instrucción es la instrucción *end*. Aunque probablemente nunca escriba un programa VHDL para una única puerta, es instructivo empezar con un ejemplo sencillo como por ejemplo una puerta AND. La declaración entity para una puerta AND de 2 entradas es:

▲ *Las comas y puntos y comas deben utilizarse de forma apropiada en todos los programas VHDL.*

```
entity AND_Gate2 is
   port (A, B: in bit; X: out bit);
end entity AND_Gate2;
```

Los términos en negrita son las palabras clave VHDL; los demás términos son identificadores que el usuario define; la sintaxis del VHDL exige el uso de paréntesis, comas y puntos y comas. Como puede ver, A y B se especifican como bits de entrada y X se especifica como un bit de salida. Los identificadores de puerto A, B y X, así como el nombre de la entidad, AND_Gate2, son definidos por el usuario y, por tanto, su nombre puede cambiarse. Como en todos los lenguajes HDL, la colocación de las comas y de los puntos y comas es crucial y deben cumplirse de forma estricta.

El elemento arquitectura (**architecture**) VHDL del programa para una puerta AND de 2 entradas descrito mediante el elemento entidad anterior es

```
architecture LogicFunction of AND_Gate2 is
begin
   X ⇐ A and B;
end architecture LogicFunction;
```

De nuevo, las palabras clave VHDL se han escrito en negrita; la sintaxis impone el uso de los puntos y comas y del símbolo ⇐. La primera instrucción de este elemento debe hacer referencia al nombre de la entidad.

La entidad y la arquitectura se combinan en un único programa VHDL para describir una puerta AND, como se ilustra en la Figura 4.45.

Escritura de expresiones booleanas en VHDL. Como hemos visto, la expresión para una puerta AND de 2 entradas, $X = AB$, en VHDL se escribe como X ⇐ A **and** B;. Cualquier expresión booleana puede escribirse utilizando las palabras clave VHDL **not, and, or, nand, nor, xor** y **xnor**. Por ejemplo, la expresión booleana $X = A + B + C$ puede escribirse en VHDL como X ⇐ A **or** B **or** C;. La expresión booleana $X = A\overline{B} + \overline{C}D$ puede escribirse en VHDL como X ⇐ (A **and not** B) **or** (**not** C **and** D);. Veamos otro ejemplo, la instrucción VHDL para una puerta NAND de 2 entradas puede escribirse como X ⇐ **not** (A **and** B); o también podría escribirse como X ⇐ A **nand** B);.

```
entity AND_Gate2 is
  port (A, B: in bit; X: out bit);
end entity AND_Gate2;

architecture LogicFunction of AND_Gate2 is
begin
  X <= A and B;
end architecture LogicFunction;
```

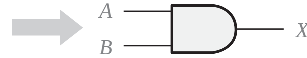


FIGURA 4.45 Un programa VHDL para una puerta AND de 2 entradas.

EJEMPLO 4.35

Escribir un programa VHDL para describir el circuito lógico de la Figura 4.46.

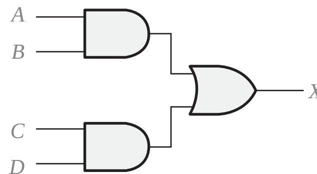


FIGURA 4.46

Solución En álgebra de Boole, este circuito lógico se describe como sigue:

$$X = AB + CD$$

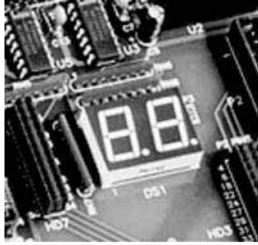
El programa VHDL es el siguiente. La entidad se denomina AND_OR.

```
entity AND_OR is
  port (A, B, C, D: in bit; X: out bit);
end entity AND_OR;
architecture LogicFunction of AND_OR is
begin
  X <= (A and B) or (C and D);
end architecture LogicFunction;
```

Problema relacionado Escribir la instrucción VHDL que describa el circuito lógico si una puerta NOR reemplaza a la puerta OR de la Figura 4.46.

REVISIÓN DE LA SECCIÓN 4.12

1. ¿Qué es el HDL?
2. Nombrar los dos elementos de diseño fundamentales en un programa VHDL.
3. ¿Para qué sirve la entidad?
4. ¿Para qué sirve la arquitectura?



APLICACIÓN A LOS SISTEMAS DIGITALES

Los displays de 7 segmentos se utilizan en toda clase de productos. El sistema de control y recuento de pastillas, que se ha descrito en el Capítulo 1, tiene dos displays de 7 segmentos. Estos displays se utilizan junto con circuitos lógicos que decodifican un número BCD y activan los dígitos adecuados del display. En esta sección, nos vamos a centrar en un diseño con un número mínimo de puertas para ilustrar las aplicaciones de las expresiones booleanas y de los mapas de Karnaugh. De forma opcional, también se aplica el lenguaje VHDL.

El display de 7 segmentos

La Figura 4.47 muestra un display común formado por siete elementos o segmentos. Excitando determinadas combinaciones de estos segmentos, se pueden obtener cada uno de los diez dígitos decimales. La Figura 4.48 muestra este tipo de display digital para cada uno de los diez dígitos, en el que se utiliza un segmento gris oscuro para indicar cuál está excitado. Para generar un 1, se excitan los segmentos *b* y *c*; para producir un 2, se excitan los segmentos *a, b, g, e* y *d*, y así sucesivamente.

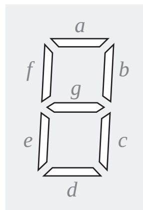


FIGURA 4.47 Disposición de los segmentos en un display de 7 segmentos.

Displays de LED. Un tipo muy común de display de 7 segmentos es el de diodos emisores de luz (*Light-Emitting Diode*, **LED**), colocados como se muestra en la Figura 4.49. Cada segmento es un LED que emite luz cuando lo atraviesa una corriente eléctrica. En la Figura 4.49(a), la configuración en ánodo común requiere un circuito de excitación, que proporcione un nivel de tensión bajo para activar un determinado segmento. Cuando se aplica un nivel BAJO a la entrada de un segmento, el LED se enciende y circula corriente a su través. En la Figura 4.49(b), la configuración en cátodo común requiere un circuito de excitación que proporcione un nivel de tensión alto para activar un cierto segmento. Cuando se aplica un nivel ALTO a la entrada del segmento, el LED se enciende y circula corriente a su través.

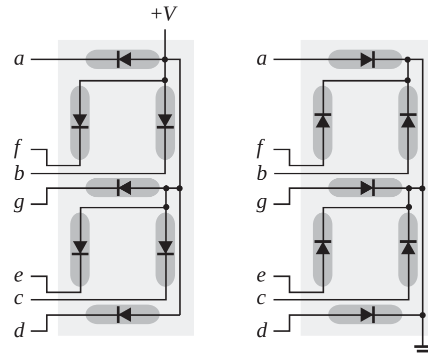


FIGURA 4.49 Configuraciones de los display de LED de 7 segmentos.

Displays de LCD. Otro tipo común de displays de 7-segmentos es el de cristal líquido, (**LCD**, *Liquid Crystal Display*). Los LCD funcionan polarizando la luz de forma que un segmento que no está activado refleja la luz incidente, por lo que se ilumina. Un segmento activado no

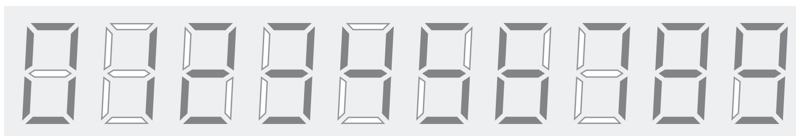


FIGURA 4.48 Display para dígitos decimales mediante un dispositivo de 7 segmentos.

refleja la luz incidente y, por tanto, permanece oscuro. Los LCD consumen mucha menos potencia que los LED, pero no se pueden ver en la oscuridad, mientras que los LED sí.

Lógica de los segmentos

Cada segmento se utiliza para varios dígitos decimales, pero ninguno de ellos se emplea para representar los diez dígitos; por tanto, cada segmento tiene que activarse mediante su propio circuito de decodificación que detecta la aparición de cualquier número en el que haya que usar ese segmento. A partir de las Figuras 4.47 y 4.48, se determinan los segmentos que hay que activar para representar cada uno de los dígitos, los cuales se enumeran en la Tabla 4.9.

Dígito	Segmentos activados
0	*a, b, c, d, e, f*
1	*b, c*
2	*a, b, d, e, g*
3	*a, b, c, d, g*
4	*b, c, f, g*
5	*a, c, d, f, g*
6	*a, c, d, e, f, g*
7	*a, b, c*
8	*a, b, c, d, e, f, g*
9	*a, b, c, d, f, g*

TABLA 4.9 Segmentos activados para cada dígito decimal.

Tabla de verdad de la lógica de segmentos. La lógica de decodificación de segmentos requiere cuatro entradas en código decimal binario (BCD) y siete salidas, una para cada segmento del display, como se indica en el diagrama de bloques de la Figura 4.50. La tabla de verdad de salida múltiple, que se muestra en la Tabla 4.10, corresponde en realidad a siete tablas de verdad, que podrían separarse en una tabla por segmento. Si aparece un 1 en las columnas de salida de la tabla, indica que el segmento está activado.

Puesto que el código BCD no incluye los valores binarios 1010, 1011, 1100, 1101, 1110 y 1111, estas combinaciones no van nunca a aparecer en las entradas y pueden, por tanto, tratarse como condiciones indiferentes (X), como se indica en la tabla de verdad. Para coincidir con la mayoría de fabricantes de circuitos integrados, en esta aplicación una A representa el bit menos significativo y una D indica el bit más significativo.

Expresiones booleanas de la lógica de segmentos. A partir de la tabla de verdad se puede escribir para cada segmento una expresión suma de productos o producto de sumas. Por ejemplo, la expresión suma de productos estándar para el segmento a es:

$$a = \bar{D}\bar{C}\bar{B}\bar{A} + \bar{D}\bar{C}B\bar{A} + \bar{D}\bar{C}BA + \bar{D}C\bar{B}A$$
$$+ \bar{D}CB\bar{A} + \bar{D}CBA + D\bar{C}\bar{B}\bar{A} + D\bar{C}\bar{B}A$$

y la expresión suma de productos estándar para el segmento e es

$$e = \bar{D}\bar{C}\bar{B}\bar{A} + \bar{D}\bar{C}B\bar{A} + \bar{D}CB\bar{A} + D\bar{C}\bar{B}\bar{A}$$

De forma similar, se pueden desarrollar expresiones para los restantes segmentos. Como se puede ver, la expresión para el segmento a consta de ocho productos y la expresión para el segmento e tienen cuatro términos productos que representan cada una de las entradas BCD que activan dicho segmento. Esto significa que la implementación de la suma de productos estándar de la lógica del segmento a requiere un circuito AND-OR formado por ocho puertas AND de 4 entradas y una puerta OR de ocho entradas. La implementación de la lógica correspondiente al segmento e requiere cuatro puertas AND de 4 entradas y una puerta OR de 4 entradas. En ambos casos, se necesitan cuatro inversores para generar el complemento de cada una de las variables.

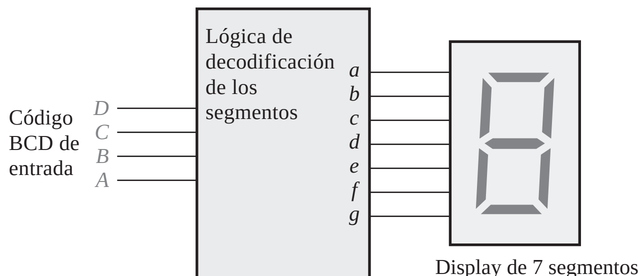


FIGURA 4.50 Diagrama de bloques de la lógica y el display de 7-segmentos.

Dígito	Entradas				Salidas de segmentos						
decimal	***D***	***C***	***B***	***A***	***a***	***b***	***c***	***d***	***e***	***f***	***g***
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1
10	1	0	1	0	X	X	X	X	X	X	X
11	1	0	1	1	X	X	X	X	X	X	X
12	1	1	0	0	X	X	X	X	X	X	X
13	1	1	0	1	X	X	X	X	X	X	X
14	1	1	1	0	X	X	X	X	X	X	X
15	1	1	1	1	X	X	X	X	X	X	X

Salida = 1 quiere decir segmento activado (encendido)
Salida = 0 quiere decir segmento desactivado (apagado)
Salida = X significa indiferente.

TABLA 4.10 Tabla de verdad para la lógica de 7 segmentos.

Minimización mediante el mapa de Karnaugh de la lógica de segmentos. Vamos a comenzar obteniendo una expresión suma de productos mínima para el segmento *a*. En la Figura 4.51 se muestra un mapa de Karnaugh correspondiente al segmento *a*. Los pasos que hay que seguir son los siguientes:

Paso 1. Los 1s de la Tabla 4.10 se pasan directamente al mapa de Karnaugh.

Paso 2. Se introducen en el mapa todas las condiciones "indiferentes" (X).

Paso 3. Se agrupan los 1s como se muestra. Se utilizan las condiciones "indiferentes" y superposiciones de celdas para conseguir los grupos más grandes posibles.

Paso 4. Se escribe el término producto mínimo para cada grupo y se suman para obtener la expresión suma de productos mínima.

No olvide que las condiciones "indiferentes" no tienen porqué incluirse en un grupo, pero en este caso se utilizan todas ellas. También hay que fijarse en que los 1s de las celdas de las esquinas se agrupan con condiciones indiferentes utilizando la adyacencia cíclica.

Implementación mínima de la lógica del segmento a. La expresión mínima suma de productos a partir del mapa de Karnaugh de la Figura 4.52 para la lógica del segmento *a* es:

$$D + B + CA + \overline{C}\overline{A}$$

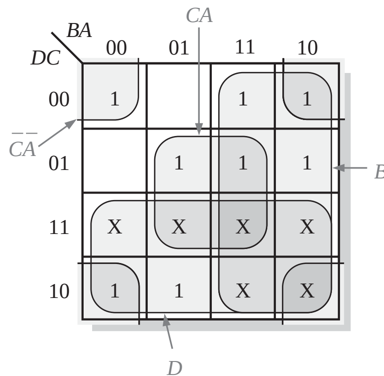
Esta expresión puede ser implementada mediante dos puertas AND de 2 entradas, una puerta OR de 4 entradas y dos inversores, como se muestra en la Figura 4.52. Compare este circuito con la implementación de la expresión estándar del segmento *a* vista anteriormente. Comprobará que el número de puertas e inversores se ha reducido de trece a cinco, disminuyendo significativamente el número de interconexiones necesarias.

La lógica mínima necesaria para cada uno de los restantes seis segmentos (*b, c, d, e, f* y *g*) puede obtenerse mediante un método similar.

Implementación VHDL (opcional)

Toda la lógica de los segmentos puede describirse utilizando VHDL para llevar a cabo la implementación en un dispositivo lógico programable. La lógica correspondiente al

Suma de productos estándar:
$\overline{D}\,\overline{C}\,\overline{B}\,\overline{A} + \overline{D}\,\overline{C}B\overline{A} + \overline{D}\,\overline{C}BA + \overline{D}C\overline{B}A + \overline{D}CB\overline{A} + \overline{D}CBA + D\overline{C}\,\overline{B}\,\overline{A} + D\overline{C}\,\overline{B}A$



Suma de productos mínima: $D + B + CA + \overline{C}\,\overline{A}$

FIGURA 4.51 Minimización de la expresión lógica del segmento *a* mediante el mapa de Karnaugh.

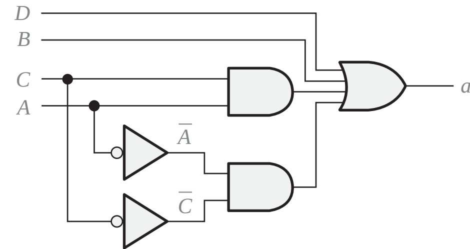


FIGURA 4.52 Implementación lógica mínima del segmento *a* de un display de 7-segmentos.

segmento *a* puede describirse mediante el siguiente programa VHDL:

```
entity SEGLOGIC is
  port (A, B, C, D: in bit; SEGa: out bit);
end entity SEGLOGIC;
architecture LogicFunction of
SEGLOGIC is
begin
  SEGa <= (A and C) or (not A
      and not C) or B or D;
end architecture LogicFunction;
```

Prácticas de sistemas

- ***Actividad 1*** Determinar la lógica mínima para el segmento *b*.
- ***Actividad 2*** Determinar la lógica mínima para el segmento *c*.
- ***Actividad 3*** Determinar la lógica mínima para el segmento *d*.
- ***Actividad 4*** Determinar la lógica mínima para el segmento *e*.
- ***Actividad 5*** Determinar la lógica mínima para el segmento *f*.
- ***Actividad 6*** Determinar la lógica mínima para el segmento *g*.
- ***Actividad opcional*** Completar el programa VHDL para los siete segmentos incluyendo en la arquitectura la descripción lógica de cada segmento.

RESUMEN

- En la Figura 4.53 se muestran los símbolos y las expresiones booleanas de salida para un inversor y puertas de dos entradas.

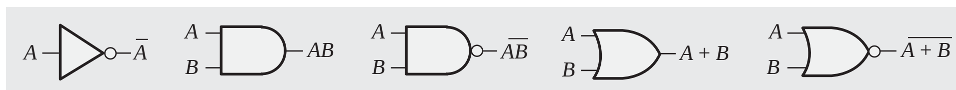


FIGURA 4.53

- Leyes conmutativas: $A + B = B + A$
  $AB = BA$

- Leyes asociativas: $A + (B + C) = (A + B) + C$
  $A(BC) = (AB)C$
- Ley distributiva: $A(B + C) = AB + AC$
- Reglas del álgebra booleana:
  1. $A + 0 = A$
  2. $A + 1 = 1$
  3. $A \cdot 0 = 0$
  4. $A \cdot 1 = A$
  5. $A + A = A$
  6. $A + \overline{A} = 1$
  7. $A \cdot A = A$
  8. $A \cdot \overline{A} = 0$
  9. $\overline{\overline{A}} = A$
  10. $A + AB = A$
  11. $A + \overline{A}B = A + B$
  12. $(A + B)(A + C) = A + BC$
- Teoremas de DeMorgan:
  1. El complemento de un producto es igual a la suma de los complementos de los términos del producto.
  $\overline{XY} = \overline{X} + \overline{Y}$
  2. El complemento de una suma es igual al producto de los complementos de los términos de la suma,
  $\overline{X + Y} = \overline{X}\,\overline{Y}$
- En la Figura 4.54 se muestran los mapas de Karnaugh para 3 y 4 variables. Un mapa de 5 variables se forma a partir de dos tablas de 4 variables.

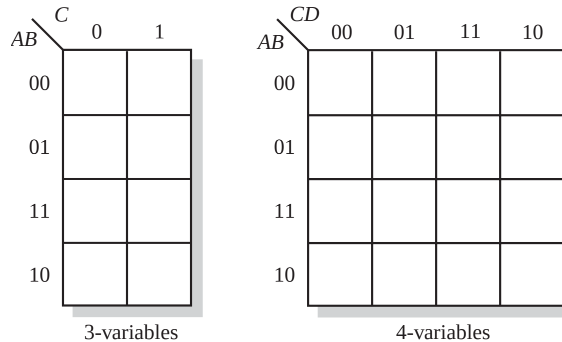


FIGURA 4.54

- El elemento de diseño básico en VHDL es una pareja entidad/arquitectura.

PALABRAS CLAVE

Las palabras clave y otros términos que se han resaltado en negrita se encuentran en el glosario final del libro.

Complemento El inverso u opuesto de un número. En el álgebra de Boole, la función inversa expresada mediante una barra sobre una variable. El complemento de 1 es 0, y viceversa.

Indiferente Combinación de literales de entrada que no pueden ocurrir y que se utilizan como 1s o 0s en un mapa de Karnaugh.

Mapa de Karnaugh Disposición de celdas que representa las combinaciones de literales en una expresión booleana y que se utiliza para la simplificación sistemática de la expresión.

Minimización El proceso que da como resultado una expresión booleana suma de productos o un producto de sumas que contiene el menor número de literales posible por término.

Producto de sumas Expresión booleana que consiste simplemente en multiplicar (operación AND) términos suma (operación OR).

Suma de productos Expresión booleana que consiste simplemente en sumar (operación OR) términos que contienen productos (operación AND).
Término producto Producto booleano de dos o más literales equivalente a una operación AND.
Término suma Suma booleana de dos o más literales equivalente a la operación OR.
Variable Símbolo utilizado para representar una magnitud lógica que puede tener tanto un valor 1 como 0; generalmente se designa mediante una letra cursiva.
VHDL Lenguaje estándar de descripción hardware. IEEE Std. 1076-1993.

AUTOTEST

Las respuestas se encuentran al final del capítulo.

1. El complemento de una variable siempre es:
(a) 0 **(b)** 1 **(c)** igual a la variable **(d)** el inverso de la variable

2. La expresión booleana $A + \overline{B} + C$ es:
(a) un término suma **(b)** un literal
(c) un término producto **(d)** un término complementado

3. La expresión booleana $A\overline{B}C\overline{D}$ es:
(a) un término suma **(b)** un término producto
(c) un literal **(d)** siempre 1

4. El dominio de la expresión $A\overline{B}CD + A\overline{B} + \overline{C}D + B$ es:
(a) A y D **(b)** Sólo B **(c)** A, B, C y D **(d)** ninguno de los anteriores

5. De acuerdo con la ley conmutativa de la suma,
(a) $AB = BA$ **(b)** $A = A + A$
(c) $A + (B + C) = (A + B) + C$ **(d)** $A + B = B + A$

6. De acuerdo con la ley asociativa de la multiplicación,
(a) $B = BB$ **(b)** $A\,(BC) = (AB)C$
(c) $A + B = B + A$ **(d)** $B + B\,(B + 0)$

7. De acuerdo con la ley distributiva,
(a) $A\,(B + C) = AB + AC$ **(b)** $A\,(BC) = ABC$
(c) $A(A + 1) = A$ **(d)** $A + AB = A$

8. ¿Cuál de las siguientes no es una regla válida del álgebra booleana?
(a) $A + 1 = 1$ **(b)** $A = \overline{A}$
(c) $AA = A$ **(d)** $A + 0 = A$

9. ¿Cuál de las siguientes reglas establece que si una entrada de una puerta AND es siempre 1, la salida es igual a la otra entrada?
(a) $A + 1 = 1$ **(b)** $A + A = A$
(c) $A \cdot A = A$ **(d)** $A \cdot 1 = A$

10. De acuerdo con los teoremas de DeMorgan, ¿cuáles de las siguientes igualdades son correctas?
(a) $\overline{AB} = \overline{A} + \overline{B}$ **(b)** $\overline{XYZ} = \overline{X} + \overline{Y} + \overline{Z}$
(c) $\overline{A + B + C} = \overline{A}\overline{B}\overline{C}$ **(d)** Todas las respuestas

11. La expresión booleana $X = AB + CD$ representa

(a) dos operaciones OR multiplicadas (AND). **(b)** Una puerta AND de 4 entradas
(c) dos operaciones AND sumadas (OR) **(d)** una operación OR-exclusiva

12. Un ejemplo de una expresión suma de productos es

(a) $A + B(C + D)$ **(b)** $\overline{A}B + A\overline{C} + A\overline{B}C$
(c) $(\overline{A} + B + C)(A + \overline{B} + C)$ **(d)** Las respuestas (a) y (b)

13. Un ejemplo de una expresión producto de sumas es

(a) $A(B + C) + A\overline{C}$ **(b)** $(A + B)(\overline{A} + B + \overline{C})$
(c) $\overline{A} + \overline{B} + BC$ **(d)** Las respuestas (a) y (b)

14. Un ejemplo de una expresión suma de productos estándar es

(a) $\overline{A}B + A\overline{B}C) + AB\overline{D}$ **(b)** $A\overline{B}C + A\overline{C}D$
(c) $A\overline{B} + \overline{A}B + AB$ **(d)** $A\overline{B}C\overline{D} + \overline{A}B + \overline{A}$

15. Un mapa de Karnaugh de 3 variables tiene

(a) ocho celdas **(b)** tres celdas **(c)** dieciséis celdas **(d)** cuatro celdas

16. En un mapa de Karnaugh de 4 variables, un término producto de dos variables se obtiene de un

(a) grupo de 2 celdas de 1s **(b)** grupo de 8 celdas de 1s
(c) grupo de 4 celdas de 1s **(d)** grupo de 4 celdas de 0s

17. En un mapa de Karnaugh, la agrupación de 0s produce

(a) una expresión producto de sumas **(b)** una expresión suma de productos
(c) una condición "indiferente" **(d)** un circuito lógico AND-OR

18. Un mapa de Karnaugh de 5 variables tiene

(a) dieciséis celdas **(b)** treinta y dos celdas **(c)** sesenta y cuatro celdas

19. Un SPLD que tiene una matriz AND programable y una matriz OR fija es una

(a) PROM **(b)** PLA **(c)** PAL **(d)** GAL

20. VHDL es un tipo de

(a) circuito lógico programable **(b)** lenguaje de descripción hardware
(c) matriz programable **(d)** matemáticas lógicas

21. En VHDL, un puerto es

(a) un tipo de entidad **(b)** un tipo de arquitectura
(c) una entrada o una salida **(d)** un tipo de variable

PROBLEMAS

Las respuestas a los problemas impares se encuentran al final del libro.

SECCIÓN 4.1 Operaciones y expresiones booleanas

1. Utilizando la notación booleana, escribir una expresión que sea 1 siempre que una o más de sus variables (A, B, C y D) sean 1.

2. Escribir una expresión que sea 1 sólo si todas sus variables (A, B, C, D y E) son 1.

3. Escribir una expresión que sea 1 cuando una o más variables (A, B y C) son 0.

4. Evaluar las siguientes operaciones:

(a) $0 + 0 + 1$ **(b)** $1 + 1 + 1$ **(c)** $1 \cdot 0 \cdot 0$

(d) $1 \cdot 1 \cdot 1$ **(e)** $1 \cdot 0 \cdot 1$ **(f)** $1 \cdot 1 + 0 \cdot 1 \cdot 1$

5. Hallar los valores de las variables que hacen que cada término producto sea 1 y que cada suma sea 0.

(a) AB **(b)** $A\bar{B}C$ **(c)** $A + B$ **(d)** $\bar{A} + B + \bar{C}$

(e) $\bar{A} + \bar{B} + C$ **(f)** $\bar{A} + B$ **(g)** $A\bar{B}\bar{C}$

6. Hallar los valores de X para todos los posibles valores de las variables.

(a) $X = (A + B)C + B$ **(b)** $X = (\overline{A + B})C$ **(c)** $X = A\bar{B}C + AB$

(e) $X = (A + B)(\bar{A} + B)$ **(f)** $X = (A + BC)(\bar{B} + \bar{C})$

SECCIÓN 4.2 Leyes y reglas del álgebra booleana

7. Identificar la ley del álgebra de Boole en que está basada cada una de las siguientes igualdades.

(a) $A\bar{B} + CD + A\bar{C}D + B = B + A\bar{B} + A\bar{C}D + CD$

(b) $AB\bar{C}D + \overline{ABC} = D\bar{C}BA + \overline{CBA}$

(c) $AB(CD + E\bar{F} + GH) = ABCD + ABE\bar{F} + ABGH$

8. Identificar la regla o reglas del álgebra de Boole en que está basada cada una de las siguientes igualdades.

(a) $\overline{\overline{AB + CD}} + \overline{EF} = AB + CD + \overline{EF}$ **(b)** $A\bar{A}B + AB\bar{C} + AB\bar{B} = AB\bar{C}$

(c) $A(BC + BC) + AC = A(BC) + AC$ **(d)** $AB(C + \bar{C}) + AC = AB + AC$

(e) $A\bar{B} + A\bar{B}C = A\bar{B}$ **(f)** $ABC + \overline{AB} + \overline{ABC}D = ABC + \overline{AB} + D$

SECCIÓN 4.3 Teoremas de DeMorgan

9. Aplicar los teoremas de DeMorgan a cada expresión:

(a) $\overline{A + \bar{B}}$ **(b)** $\overline{\bar{A}B}$ **(c)** $\overline{A + B + C}$ **(d)** $\overline{ABC}$

(e) $\overline{A(B + C)}$ **(f)** $\overline{AB} + \overline{CD}$ **(g)** $\overline{AB + CD}$ **(h)** $\overline{(A + \bar{B})(\bar{C} + D)}$

10. Aplicar los teoremas de DeMorgan a cada expresión:

(a) $\overline{A\bar{B}(C + \bar{D})}$ **(b)** $\overline{AB(CD + EF)}$

(c) $\overline{(A + \bar{B} + C + \bar{D})} + \overline{ABC\bar{D}}$ **(d)** $\overline{\overline{(\bar{A} + B + C + D)}\,\overline{(A\bar{B}\bar{C}D)}}$

(e) $\overline{\overline{AB}(CD + \bar{E}F)(\overline{AB} + \overline{CD})}$

11. Aplicar los teoremas de DeMorgan a las siguientes expresiones:

(a) $\overline{\overline{\overline{(\overline{ABC})(\overline{EFG})} + \overline{(\overline{HIJ})(\overline{KLM})}}}$ **(b)** $\overline{(A + \overline{B\bar{\bar{C}}} + CD)} + \overline{\overline{BC}}$

(c) $\overline{\overline{\overline{(A + B)}\,\overline{(C + D)}\,\overline{(E + F)}\,\overline{(G + H)}}}$

SECCIÓN 4.4 Análisis booleano de los circuitos lógicos

12. Escribir la expresión booleana para cada puerta lógica de la Figura 4.55.

13. Escribir la expresión booleana para cada uno de los circuitos lógicos de la Figura 4.56.

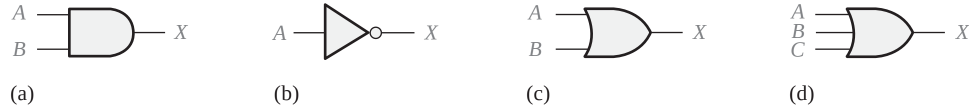


FIGURA 4.55

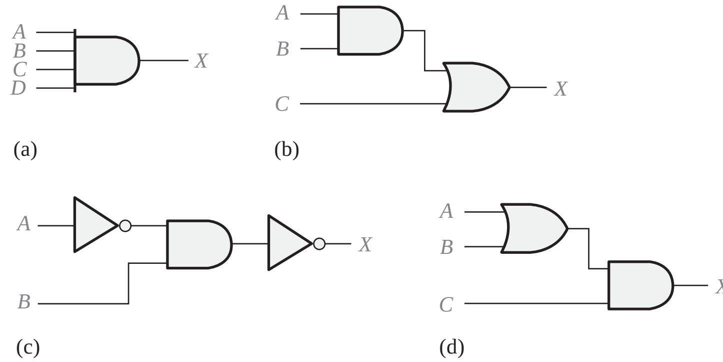


FIGURA 4.56

14. Dibujar el circuito lógico representado por cada una de las siguientes expresiones.

(a) $A + B + C$ **(b)** ABC **(c)** $AB + C$ **(d)** $AB + CD$

15. Dibujar el circuito lógico representado por cada una de las siguientes expresiones.

(a) $A\overline{B}+\overline{A}B$ **(b)** $AB+\overline{A}\overline{B}+\overline{A}BC$
(c) $\overline{A}B(C+\overline{D})$ **(d)** $A+(B[C+D(B+\overline{C})]$

16. Construir una tabla de verdad para cada una de las siguientes expresiones booleanas.

(a) $A+B$ **(b)** AB **(c)** $AB+BC$
(d) $(A+B)C$ **(e)** $(A+B)(\overline{B}+C)$

SECCIÓN 4.5 Simplificación mediante el álgebra de Boole

17. Mediante las técnicas del álgebra de Boole, simplificar las siguientes expresiones lo máximo posible:

(a) $A(A+B)$ **(b)** $A(\overline{A}+AB)$ **(c)** $BC+\overline{B}C$
(d) $A(A+\overline{A}B)$ **(e)** $A\overline{B}C+\overline{A}BC+\overline{A}\overline{B}C$

18. Mediante las técnicas del álgebra de Boole, simplificar las siguientes expresiones:

(a) $(A+\overline{B})(A+C)$ **(b)** $\overline{A}B+\overline{A}B\overline{C}+\overline{A}BCD+\overline{A}B\overline{C}\overline{D}E$
(c) $AB+\overline{AB}C+A$ **(d)** $(A+\overline{A})(AB+AB\overline{C})$
(e) $AB+(\overline{A}+\overline{B})C+AB$

19. Mediante las técnicas del álgebra de Boole, simplificar las siguientes expresiones:

(a) $BD+B(D+E)+\overline{D}(D+F)$ **(b)** $\overline{A}\overline{B}C+\overline{(A+B+\overline{C})}+\overline{A}\overline{B}\overline{C}D$
(c) $(B+BC)(B+\overline{B}C)(B+D)$ **(d)** $ABCD+AB(\overline{CD})+(\overline{AB})CD$
(e) $ABC[AB+\overline{C}(BC+AC)]$

20. Determinar cuáles de los circuitos lógicos de la Figura 4.57 son equivalentes.

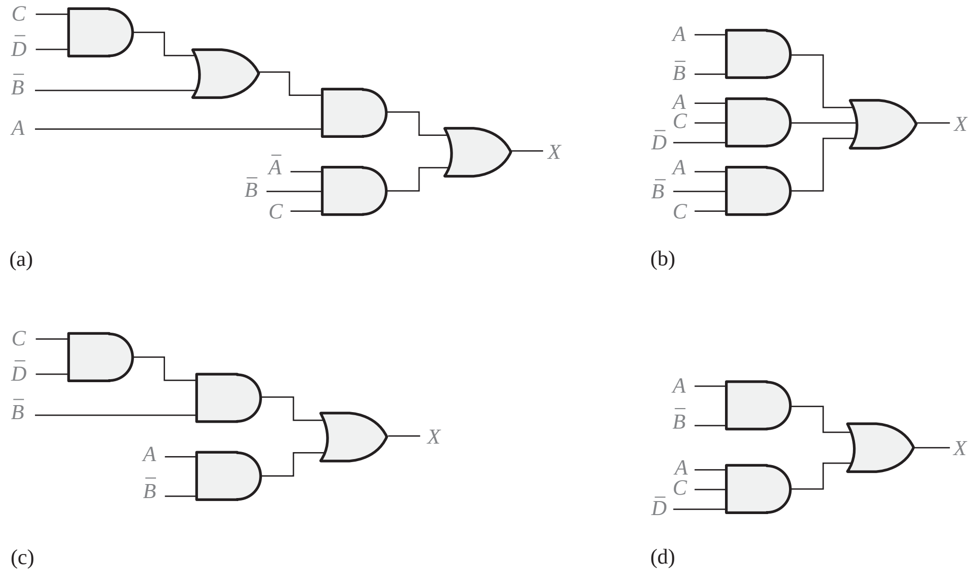


FIGURA 4.57

SECCIÓN 4.6 Formas estándar de las expresiones booleanas

21. Convertir las siguientes expresiones en sumas de productos:

(a) $(A+B)(C+\overline{B})$ **(b)** $(A+\overline{B}C)C$ **(c)** $(A+C)(AB+AC)$

22. Convertir las siguientes expresiones en sumas de productos:

(a) $AB+CD(A\overline{B}+CD)$ **(b)** $AB(\overline{B}\overline{C}+BD)$ **(c)** $A+B[AC+(B+\overline{C})D]$

23. Definir el dominio de cada suma de productos del Problema 21 y convertir la expresión a su forma estándar.

24. Convertir cada suma de productos del Problema 22 a su forma estándar.

25. Determinar el valor binario de cada término en las expresiones suma de productos del Problema 23.

26. Determinar el valor binario de cada término en las expresiones suma de productos del Problema 24.

27. Convertir cada una de las expresiones suma de productos estándar del Problema 23 a su forma producto de sumas estándar.

28. Convertir cada una de las expresiones suma de productos estándar del Problema 24 a su forma producto de sumas estándar.

SECCIÓN 4.7 Expresiones booleanas y tablas de verdad

29. Desarrollar la tabla de verdad de cada una de las siguientes expresiones suma de productos estándar:

(a) $A\overline{B}C+\overline{A}B\overline{C}+ABC$ **(b)** $\overline{XYZ}+\overline{X}\overline{Y}Z+XY\overline{Z}+X\overline{Y}Z+\overline{X}YZ$

30. Desarrollar la tabla de verdad de cada una de las siguientes expresiones suma de productos estándar:

(a) $\overline{A}B\overline{C}D+\overline{A}BC\overline{D}+A\overline{B}\overline{C}D+\overline{A}\overline{B}\overline{C}\overline{D}$

(b) $WXYZ+WXY\overline{Z}+\overline{W}XYZ+W\overline{X}YZ+WX\overline{Y}Z$

31. Desarrollar la tabla de verdad de cada una de las siguientes expresiones suma de productos estándar:

(a) $\overline{A}B + AB\overline{C} + \overline{A}\overline{C} + A\overline{B}C$ **(b)** $\overline{X} + Y\overline{Z} + WZ + X\overline{Y}Z$

32. Desarrollar la tabla de verdad de cada una de las siguientes expresiones producto de sumas estándar:

(a) $(\overline{A}+\overline{B}+\overline{C})(A+B+C)(A+\overline{B}+C)$

(b) $(\overline{A}+B+\overline{C}+D)(A+\overline{B}+C+\overline{D})(A+\overline{B}+\overline{C}+D)(\overline{A}+B+C+\overline{D})$

33. Desarrollar la tabla de verdad de cada una de las siguientes expresiones producto de sumas estándar:

(a) $(A+B)(A+C)(A+B+C)$

(b) $(A+\overline{B})(A+\overline{B}+\overline{C})(B+C+\overline{D})(\overline{A}+B+\overline{C}+D)$

34. Para cada tabla de verdad de la Figura 4.58, obtener una expresión suma de productos estándar y un producto de sumas estándar.

(a)

ABC	*X*
0 0 0	0
0 0 1	1
0 1 0	0
0 1 1	0
1 0 0	1
1 0 1	1
1 1 0	0
1 1 1	1

(b)

ABC	*X*
0 0 0	0
0 0 1	0
0 1 0	0
0 1 1	0
1 0 0	0
1 0 1	1
1 1 0	1
1 1 1	1

(c)

ABCD	*X*
0 0 0 0	1
0 0 0 1	1
0 0 1 0	0
0 0 1 1	1
0 1 0 0	0
0 1 0 1	1
0 1 1 0	1
0 1 1 1	0
1 0 0 0	0
1 0 0 1	1
1 0 1 0	0
1 0 1 1	0
1 1 0 0	1
1 1 0 1	0
1 1 1 0	0
1 1 1 1	0

(d)

ABCD	*X*
0 0 0 0	0
0 0 0 1	0
0 0 1 0	1
0 0 1 1	0
0 1 0 0	1
0 1 0 1	1
0 1 1 0	0
0 1 1 1	1
1 0 0 0	0
1 0 0 1	0
1 0 1 0	0
1 0 1 1	1
1 1 0 0	1
1 1 0 1	0
1 1 1 0	0
1 1 1 1	1

FIGURA 4.58

SECCIÓN 4.8 Mapas de Karnaugh

35. Dibujar un mapa de Karnaugh de 3 variables y etiquetar cada celda según su valor binario.

36. Dibujar un mapa de Karnaugh de 4 variables y etiquetar cada celda según su valor binario.

37. Escribir los términos producto estándar correspondientes a cada celda de un mapa de Karnaugh de 3 variables.

SECCIÓN 4.9 Minimización de una suma de productos mediante el mapa de Karnaugh

38. Utilizar un mapa de Karnaugh para hallar la suma de productos mínima para cada una de las expresiones siguientes.

(a) $\overline{A}\overline{B}\overline{C} + \overline{A}\overline{B}C + A\overline{B}C$ **(b)** $AC(\overline{B}+C)$

(c) $\bar{A}(BC + B\bar{C}) + A(BC + B\bar{C})$ (d) $\bar{A}\bar{B}\bar{C} + A\bar{B}\bar{C} + \bar{A}B\bar{C} + AB\bar{C}$

39. Utilizar un mapa de Karnaugh para simplificar las expresiones siguientes a su forma suma de productos mínima.

(a) $\bar{A}\bar{B}\bar{C} + A\bar{B}C + \bar{A}BC + AB\bar{C}$ (b) $AC[\bar{B} + B(B + \bar{C})]$
(c) $DE\bar{F} + \bar{D}E\bar{F} + \bar{D}\bar{E}\bar{F}$

40. Expandir las expresiones siguientes a su forma suma de productos estándar.

(a) $AB + A\bar{B}C + ABC$ (b) $A + BC$
(c) $A\bar{B}\bar{C}D + AC\bar{D} + B\bar{C}D + \bar{A}BC\bar{D}$ (d) $A\bar{B} + A\bar{B}\bar{C}D + CD + B\bar{C}D + ABCD$

41. Minimizar las expresiones del Problema 40 utilizando un mapa de Karnaugh.

42. Utilizar un mapa de Karnaugh para reducir las expresiones siguientes a su forma suma de productos mínima.

(a) $A + B\bar{C} + CD$
(b) $\bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}\bar{B}\bar{C}D + ABCD + ABC\bar{D}$
(c) $\bar{A}B(\bar{C}\bar{D} + \bar{C}D) + AB(\bar{C}\bar{D} + \bar{C}D) + A\bar{B}\bar{C}D$
(d) $(\bar{A}\bar{B} + A\bar{B})(CD + C\bar{D})$
(e) $\bar{A}\bar{B} + A\bar{B} + \bar{C}\bar{D} + C\bar{D}$

43. Reducir la función especificada en la tabla de verdad de la Figura 4.59 a su forma suma de productos mínima mediante un mapa de Karnaugh.

44. Utilizar el mapa de Karnaugh para implementar la forma suma de productos mínima de la función lógica especificada en la tabla de verdad de la Figura 4.60.

45. Resolver el Problema 44 para una situación en que las seis últimas combinaciones binarias no están permitidas.

Entradas			Salida
A	*B*	*C*	*X*
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

FIGURA 4.59

Entradas				Salida
A	*B*	*C*	*D*	*X*
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	0
1	1	1	1	1

FIGURA 4.60

SECCIÓN 4.10 Minimización de un producto de sumas mediante el mapa de Karnaugh

46. Utilizar un mapa de Karnaugh para hallar la suma de productos mínima de las siguientes expresiones:

(a) $(A+B+C)(\overline{A}+\overline{B}+\overline{C})(A+\overline{B}+C)$

(b) $(X+\overline{Y})(\overline{X}+Z)(X+\overline{Y}+\overline{Z})(\overline{X}+\overline{Y}+Z)$

(c) $A(B+\overline{C})(\overline{A}+C)(A+\overline{B}+C)(\overline{A}+B+\overline{C})$

47. Utilizar un mapa de Karnaugh para simplificar las siguientes expresiones a su forma producto de sumas mínima:

(a) $(A+\overline{B}+C+\overline{D})(\overline{A}+B+\overline{C}+D)(\overline{A}+\overline{B}+\overline{C}+\overline{D})$

(b) $(X+\overline{Y})(W+\overline{Z})(\overline{X}+\overline{Y}+\overline{Z})(W+X+Y+Z)$

48. Para la función especificada en la tabla de verdad de la Figura 4.59, determinar el producto de sumas mínimo mediante el mapa de Karnaugh.

49. Determinar el producto de sumas mínimo para la función de la tabla de verdad de la Figura 4.60.

50. Convertir cada una de las siguientes expresiones producto de sumas mínimo a la forma de suma de productos mínima utilizando un mapa de Karnaugh.

(a) $(A+\overline{B})(A+\overline{C})(\overline{A}+\overline{B}+C)$

(b) $(\overline{A}+B)(\overline{A}+\overline{B}+\overline{C})(B+\overline{C}+D)(A+\overline{B}+C+\overline{D})$

SECCIÓN 4.11 Mapa de Karnaugh de cinco variables

51. Minimizar la siguiente suma de productos utilizando un mapa de Karnaugh.

$$X = \overline{A}B\overline{C}D\overline{E} + \overline{A}\,\overline{B}\,\overline{C}DE + A\overline{B}\,\overline{C}DE + AB\overline{C}\,\overline{D}\,\overline{E} + \overline{A}BCD\overline{E} + \overline{A}BC\overline{D}E$$
$$+ \overline{A}\,\overline{B}\,\overline{C}\,\overline{D}\,\overline{E} + \overline{A}\,\overline{B}CDE + AB\overline{C}D\overline{E} + AB\overline{C}DE$$

52. Aplicar el mapa de Karnaugh para minimizar la siguiente suma de productos.

$$A = \overline{V}WXYZ + V\overline{W}XYZ + VW\overline{X}YZ + VWX\overline{Y}Z + VWXY\overline{Z} + \overline{V}\,\overline{W}\,\overline{X}\,\overline{Y}\,\overline{Z} + \overline{V}\,\overline{W}\,\overline{X}Y\overline{Z} + \overline{V}\,\overline{W}X\overline{Y}\,\overline{Z} + \overline{V}W\overline{X}\,\overline{Y}\,\overline{Z}$$

SECCIÓN 4.12 VHDL (opcional)

53. Escribir un programa VHDL para el circuito lógico de la Figura 4.61.

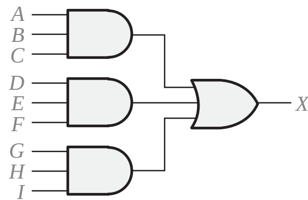
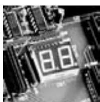


FIGURA 4.61

54. Escribir un programa VHDL para la expresión

$$Y = A\overline{B}C + \overline{A}\,\overline{B}C + A\overline{B}\,\overline{C} + \overline{A}BC$$



Aplicación a los sistemas digitales

55. Si es necesario elegir un tipo de display para trabajar bajo condiciones de baja luminosidad, ¿cuál se seleccionaría, un display de 7 segmentos de diodos LED o de cristal líquido? ¿Por qué?

56. Explicar por qué los códigos 1010, 1011, 1100, 1101, 1110 y 1111 pertenecen a la categoría de condiciones "indiferentes" en las aplicaciones con displays de 7 segmentos.

57. Para el segmento *b*, ¿cuántas puertas e inversores menos se necesitan para implementar la suma de productos mínima con respecto a la suma de productos estándar?

58. Repetir el Problema 57 para la lógica de los segmentos *c* hasta *g*.



Problemas especiales de diseño

59. La lógica del segmento *a* de la Figura 4.52 produce una salida a nivel ALTO para activar el segmento, ocurriendo lo mismo para el resto de los segmentos. Si se utiliza un display de 7-segmentos que requiere un nivel BAJO para activar cada segmento, modificar adecuadamente la lógica del segmento.

60. Rediseñar la lógica del segmento *a* utilizando un producto de sumas mínimo ¿Cuál es más sencilla, la suma de productos mínima o el producto de sumas mínimo?

61. Repetir el Problema 60 para los segmentos *b* hasta *g*.

62. Resumir los resultados del rediseño que se ha hecho en los Problemas 60 y 61, y recomendar el mejor diseño en función del mínimo número de circuitos integrados. Especificar los tipos de CI que se utilizarían.

RESPUESTAS

REVISIONES DE CADA SECCIÓN

SECCIÓN 4.1 **Operaciones y expresiones booleanas**

1. $\overline{A} = \overline{0} = 1$ **2.** $A = 1, B = 1, C = 0;\ \overline{A} + \overline{B} + C = \overline{1} + \overline{1} + 0 = 0 + 0 + 0 = 0$

3. $A = 1, B = 0, C = 1;\ A\overline{B}C = 1 \cdot \overline{0} \cdot 1 = 1 \cdot 1 \cdot 1 = 1$

SECCIÓN 4.2 **Leyes y reglas del álgebra booleana**

1. $A + (B + C + D) = (A + B + C) + D$ **2.** $A(B + C + D) = AB + AC + AD$

SECCIÓN 4.3 **Teoremas de DeMorgan**

1. (a) $\overline{ABC} + \overline{(\overline{D} + E)} = \overline{A} + \overline{B} + \overline{C} + D\overline{E}$ **(b)** $\overline{(A + B)C} = \overline{A}\overline{B} + \overline{C}$

(c) $\overline{A + B + C} + \overline{\overline{D}E} = \overline{A}\overline{B}\overline{C} + D + \overline{E}$

SECCIÓN 4.4 **Análisis booleano de los circuitos lógicos**

1. $(C + D)B + A$

2. Tabla de verdad abreviada: la expresión es 1 cuando *A* es 1 o cuando *B* y *C* son 1, o cuando *B* y *D* son 1. La expresión es 0 para todas las demás combinaciones.

SECCIÓN 4.5 **Simplificación mediante el álgebra de Boole**

1. **1. (a)** $A + AB + A\overline{B}C = A$ **(b)** $(\overline{A} + B)C + ABC = C(\overline{A} + B)$

(c) $A\overline{B}C(BD + CDE) + A\overline{C} = A(\overline{C} + \overline{B}DE)$

2. **(a)** *Original:* 2 puertas AND, 1 puerta OR, 1 inversor. *Simplificada:* sin puertas (conexión directa).

(b) *Original:* 2 puertas OR, 2 puertas AND, 1 inversor. *Simplificada:* 1 puerta OR, 1 puerta AND, 1 inversor.

(c) *Original:* 5 puertas AND, 2 puertas OR, 2 inversores. *Simplificada:* 1 puertas AND, 1 puerta OR, 2 inversores.

SECCIÓN 4.6 Formas estándar de las expresiones booleanas

1. **(a)** Suma de productos **(b)** Producto de sumas estándar
(c) Suma de productos estándar **(d)** Producto de sumas

2. **(a)** $AB\overline{C}\overline{D}+AB\overline{C}D+ABC\overline{D}+ABCD+\overline{A}B\overline{C}D+\overline{A}BCD+\overline{A}\overline{B}C\overline{D}+\overline{A}BC\overline{D}$
(c) Ya está en forma estándar

3. **(b)** Ya está en forma estándar
(d) $(A+\overline{B}+\overline{C})(A+\overline{B}+C)(A+B+\overline{C})(A+B+C)$

SECCIÓN 4.7 Expresiones booleanas y tablas de verdad

1. $2^5=32$

2. $0110 \rightarrow \overline{W}XY\overline{Z}$

3. $1100 \rightarrow \overline{W}+\overline{X}+Y+Z$

SECCIÓN 4.8 Mapas de Karnaugh

1. **(a)** celda superior izquierda: 000 **(b)** celda inferior derecha: 101
(c) celda inferior izquierda: 100 **(d)** celda superior derecha: 001

2. **(a)** celda superior izquierda: $\overline{X}\overline{Y}\overline{Z}$ **(b)** celda inferior derecha: $X\overline{Y}Z$
(c) celda inferior izquierda: $X\overline{Y}\overline{Z}$ **(d)** celda superior derecha: $\overline{X}\overline{Y}Z$

3. **(a)** celda superior izquierda: 0000 **(b)** celda inferior derecha: 1010
(c) celda inferior izquierda: 1000 **(d)** celda superior derecha: 0010

4. **(a)** celda superior izquierda: $\overline{W}\overline{X}\overline{Y}\overline{Z}$ **(b)** celda inferior derecha: $W\overline{X}Y\overline{Z}$
(c) celda inferior izquierda: $W\overline{X}\overline{Y}\overline{Z}$ **(d)** celda superior derecha: $\overline{W}\overline{X}Y\overline{Z}$

SECCIÓN 4.9 Minimización de una suma de productos mediante el mapa de Karnaugh

1. Mapa de 8 celdas para 3 variables; mapa de 16 celdas para 4 variables.

2. $AB+B\overline{C}+\overline{A}\overline{B}C$

3. **(a)** $\overline{A}\overline{B}\overline{C}+\overline{A}BC+ABC+AB\overline{C}$
(b) $\overline{A}\overline{B}\overline{C}+\overline{A}\overline{B}C+\overline{A}B\overline{C}+ABC+A\overline{B}\overline{C}+A\overline{B}C$
(c) $\overline{A}\overline{B}\overline{C}\overline{D}+\overline{A}\overline{B}\overline{C}D+\overline{A}B\overline{C}\overline{D}+\overline{A}B\overline{C}D+\overline{A}BCD+\overline{A}BC\overline{D}+A\overline{B}\overline{C}D+A\overline{B}CD$
(d) $\overline{A}\overline{B}\overline{C}\overline{D}+\overline{A}B\overline{C}\overline{D}+AB\overline{C}\overline{D}+A\overline{B}\overline{C}\overline{D}+\overline{A}B\overline{C}D+AB\overline{C}D+A\overline{B}CD+\overline{A}\overline{B}C\overline{D}+$
$\overline{A}BC\overline{D}+ABC\overline{D}+A\overline{B}C\overline{D}$

SECCIÓN 4.10 Minimización de un producto de sumas mediante el mapa de Karnaugh

1. Cuando se pasa a un mapa de Karnaugh un producto de sumas, los 0s se colocan en las celdas cuyos valores hacen que el término suma estándar sea cero; cuando se pasa a un mapa de Karnaugh una suma de productos, los 1s se colocan en la celdas que tienen los mismos valores que los términos producto.

2. 0 en la celda 1011: $\overline{A}+B+\overline{C}+\overline{D}$ **3.** 1 en la celda 0010: $\overline{A}\overline{B}C\overline{D}$

SECCIÓN 4.11 Mapas de Karnaugh de cinco variables

1. Existen 32 combinaciones de las 5 variables ($2^5=32$).

2. $X=1$, ya que la función es 1 para todas las posibles combinaciones de las 5 variables.

SECCIÓN 4.12 **VHDL (opcional)**

1. HDL es un lenguaje de descripción hardware para dispositivos lógicos programables.
2. Entidad y arquitectura.
3. La entidad especifica las entradas y las salidas de una función lógica.
4. La arquitectura especifica la operación de una función lógica.

PROBLEMAS RELACIONADOS

4.1 $\overline{A} + B = 0$ cuando $A = 1$ y $B = 0$.

4.2 $\overline{A}\overline{B} = 1$ cuando $A = 0$ y $B = 0$.

4.3 XYZ

4.4 $W + X + Y + Z$

4.5 $ABC\overline{D}\overline{E}$

4.6 $(A + \overline{B} + \overline{C}D)\overline{E}$

4.7 $\overline{ABCD} = \overline{A} + \overline{B} + \overline{C} + \overline{D}$

4.8 $A\overline{B}$

4.9 CD

4.10 $AB\overline{C} + \overline{A}C + \overline{A}\overline{B}$

4.11 $\overline{A} + \overline{B} + \overline{C}$

4.12 $\overline{A}B\overline{C} + AB + A\overline{C} + A\overline{B} + \overline{B}\overline{C}$

4.13 $W\overline{X}YZ + W\overline{X}Y\overline{Z} + W\overline{X}Y\overline{Z} + \overline{W}\overline{X}Y\overline{Z} + WX\overline{Y}Z + WX\overline{Y}\overline{Z}$

4.14 011, 101, 110, 010, 111. Sí

4.15 $(A + \overline{B} + C)(A + \overline{B} + \overline{C})(A + B + C)(\overline{A} + B + C)$

4.16 010, 100, 001, 111, 011. Sí

4.17 Las expresiones suma de productos y producto de sumas son equivalentes.

4.18 *Véase* la Tabla 4.11

4.19 *Véase* la Tabla 4.12.

A	*B*	*C*	*X*
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

TABLA 4.11

A	*B*	*C*	*X*
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

TABLA 4.12

4.20 Las expresiones suma de productos y producto de sumas son equivalentes.

4.21 *Véase* la Figura 4.62.

AB \ C	0	1
00		
01		1
11		
10	1	1

FIGURA 4.62

AB \ CD	00	01	11	10
00				
01				1
11	1		1	1
10				

FIGURA 4.63

4.22 *Véase* la Figura 4.63.

4.23 *Véase* la Figura 4.64.

4.24 *Véase* la Figura 4.65.

AB \ C	0	1
00	1	
01	1	1
11		1
10		

FIGURA 4.64

AB \ CD	00	01	11	10
00		1		
01		1		1
11	1	1	1	1
10	1	1	1	1

FIGURA 4.65

4.25 Ninguna otra forma.

4.26 $X = B + \bar{A}C + A\bar{C}D + C\bar{D}$

4.27 $X = \bar{D} + A\bar{B}C + B\bar{C} + \bar{A}B$

4.28 $Q = X + Y$

4.29 $Q = \bar{X}\bar{Y}\bar{Z} + W\bar{X}Z + \bar{W}YZ$

4.30 *Véase* la Figura 4.66.

AB \ CD	00	01	11	10
00	0	0		
01				0
11				
10				0

FIGURA 4.66

4.31 $Q = (X + \overline{Y})(X + \overline{Z})(\overline{X} + Y + Z)$

4.32 $Q = (\overline{X} + \overline{Y} + Z)(\overline{W} + \overline{X} + Z)(W + X + Y + Z)(W + \overline{X} + Y + \overline{Z})$

4.33 $Q = \overline{Y}\overline{Z} + \overline{X}\overline{Z} + \overline{W}Y + \overline{X}\overline{Y}Z$

4.34 $Y = \overline{D}\overline{E} + \overline{A}\overline{E} + \overline{B}\overline{C}\overline{E}$

4.35 X ⇐ (A **and** B)(C **and** D);

AUTOTEST

1. (d) **2.** (a) **3.** (b) **4.** (c) **5.** (d) **6.** (b) **7.** (a) **8.** (b)
9. (d) **10.** (d) **11.** (c) **12.** (b) **13.** (b) **14.** (c) **15.** (a) **16.** (c)
17. (a) **18.** (b) **19.** (c) **20.** (b) **21.** (c)

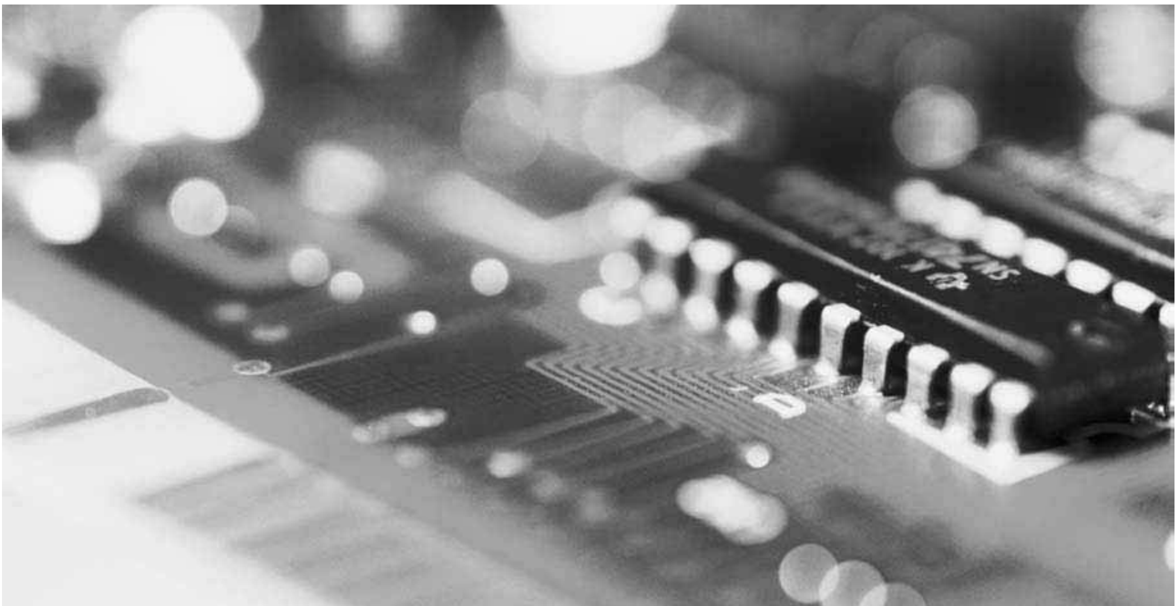
5

ANÁLISIS DE LA LÓGICA COMBINACIONAL

CONTENIDO DEL CAPÍTULO

OBJETIVOS DEL CAPÍTULO

- Analizar los circuitos lógicos combinacionales básicos, tales como AND-OR, AND-OR-Inversor, OR-exclusiva y NOR-exclusiva.
- Utilizar los circuitos AND-OR y AND-OR-Inversor para implementar expresiones como suma de productos y producto de sumas.
- Escribir la expresión booleana de salida de cualquier circuito lógico combinacional.
- Desarrollar la tabla de verdad a partir de la expresión de salida de un circuito lógico combinacional.



- Utilizar el mapa de Karnaugh para expandir una expresión de salida que contenga variables suprimidas en una suma de productos completa.
- Diseñar un circuito lógico combinacional para una expresión booleana de salida dada.
- Diseñar un circuito lógico combinacional para una tabla de verdad dada.
- Simplificar un circuito lógico combinacional a su forma mínima.
- Utilizar puertas NAND para implementar cualquier función lógica combinacional.
- Utilizar puertas NOR para implementar cualquier función lógica combinacional.
- Escribir programas VHDL para circuitos lógicos simples.
- Localizar fallos en los circuitos lógicos.
- Localizar fallos en los circuitos lógicos utilizando el seguimiento de señales y el análisis de las formas de onda.

PALABRAS CLAVE

- Puerta universal
- negativa-OR
- negativa-AND
- Componente
- Señal
- Nodo
- Seguimiento de señales

INTRODUCCIÓN

En los Capítulos 3 y 4, nos hemos ocupado de las puertas lógicas como elementos individuales y en sencillas combinaciones. Se han introducido las implementaciones de suma de productos y producto de sumas, que son las formas básicas de la lógica combinacional. Cuando se conectan puertas lógicas entre sí, con el fin de generar una determinada salida específica para determinadas combinaciones específicas de las variables de entrada, sin que haya implicado almacenamiento, el circuito resultante se califica como **lógica combinacional**. En la lógica combinacional, el nivel de salida depende siempre de la combinación de los niveles de entrada. Este capítulo amplía el material presentado en los capítulos anteriores y cubre el análisis, diseño y localización de fallos de diversos circuitos lógicos combinacionales. Se presenta el método estructural del VHDL y se aplica a la lógica combinacional.

■ ■ ■ APLICACIÓN A LOS SISTEMAS DIGITALES

Esta aplicación a un sistema digital ilustra los conceptos que se enseñan en este capítulo, mostrando cómo se puede usar la lógica combinacional para un propósito específico en una aplicación práctica. Se emplea un circuito lógico para controlar el nivel y la temperatura de un fluido en un tanque de almacenamiento. Manipulando las válvulas interna y externa, se controla la entrada y salida de flujo basándose en la información proporciona por las entradas del sensor de nivel. De forma opcional, también se explica cómo usar VHDL para describir la lógica.

5.1 CIRCUITOS LÓGICOS COMBINACIONALES BÁSICOS

En el Capítulo 4, ha aprendido que las expresiones suma de productos se implementan con una puerta AND para cada término producto y una puerta OR para sumar todos los términos producto. Esta implementación de la suma de productos se llama lógica AND-OR y es la forma básica para realizar las funciones estándar booleanas. En esta sección, se estudian las combinaciones AND-OR y AND-OR-Inversor y las puertas OR-exclusiva y NOR-exclusiva, que realmente son una forma de la lógica AND-OR.

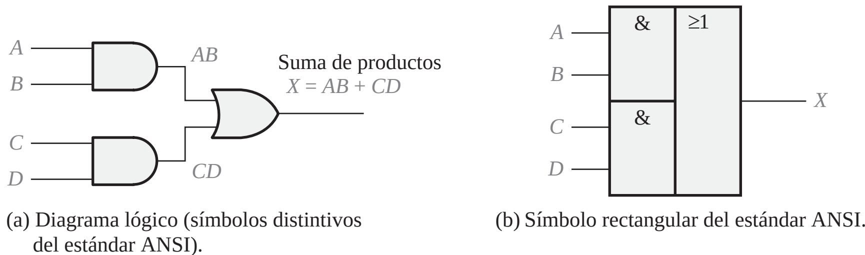
Al finalizar esta sección, el lector deberá ser capaz de:

■ Analizar y aplicar los circuitos AND-OR. ■ Analizar y aplicar los circuitos AND-OR-Inversor. ■ Analizar y aplicar las puertas OR-exclusiva. ■ Analizar y aplicar las puertas NOR-exclusiva.

Lógica AND-OR

▲ *La lógica AND-OR genera expresiones suma de productos.*

La Figura 5.1(a) muestra un circuito AND-OR formado por dos puertas AND de dos entradas y una puerta OR de dos entradas. La Figura 5.1(b) corresponde al símbolo estándar rectangular ANSI. En el diagrama se indica la expresión booleana para las salidas de las puertas AND y la suma de productos resultante para la salida *X*. En general, un circuito AND-OR puede tener cualquier número de puertas AND, cada una de ellas con cualquier número de entradas.



(a) Diagrama lógico (símbolos distintivos del estándar ANSI).

(b) Símbolo rectangular del estándar ANSI.

FIGURA 5.1 Ejemplo de circuito lógico AND-OR.

En la Tabla 5.1 se presenta la tabla de verdad para el circuito lógico AND-OR de cuatro entradas. Las salidas de la puerta AND intermedia (las columnas *AB* y *CD*) también se indican en dicha tabla.

Un circuito AND-OR implementa directamente una suma de productos, suponiendo que se dispone de los complementos de las variables. La operación lógica del circuito AND-OR de la Figura 5.1 se enuncia así:

En un circuito lógico AND-OR de 4 entradas, la salida *X* es un nivel ALTO (1) sólo si las dos entradas *A* y *B* están a nivel ALTO (1) o si las dos entradas *C* y *D* están a nivel ALTO (1).

Circuito lógico AND-OR-Inversor

Cuando se complementa (invierte) la salida de un circuito AND-OR, se obtiene el circuito AND-OR-Inversor. Recuerde que el circuito AND-OR implementa directamente la suma de productos. El producto de sumas puede implementarse con un circuito lógico AND-OR-Inversor. Esto se ilustra de la forma siguiente, partiendo del producto de sumas y desarrollando la expresión AND-OR-Inversor correspondiente.

$$X = (\overline{A} + \overline{B})(\overline{C} + \overline{D}) = (\overline{AB})(\overline{CD}) = \overline{\overline{(\overline{AB})(\overline{CD})}} = \overline{\overline{\overline{AB}} + \overline{\overline{CD}}} = \overline{AB + CD}$$

Entradas						Salida
A	*B*	*C*	*D*	*AB*	*CD*	*X*
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	0	1	0	0	0	0
0	0	1	1	0	1	1
0	1	0	0	0	0	0
0	1	0	1	0	0	0
0	1	1	0	0	0	0
0	1	1	1	0	1	1
1	0	0	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	0	0	0
1	0	1	1	0	1	1
1	1	0	0	1	0	1
1	1	0	1	1	0	1
1	1	1	0	1	0	1
1	1	1	1	1	1	1

TABLA 5.1 Tabla de verdad para el circuito lógico AND-OR de la Figura 5.1.

EJEMPLO 5.1

En una determinada planta de procesamiento químico se emplea un elemento químico líquido en un proceso de fabricación. Dicho elemento químico se almacena en tres tanques diferentes. Un sensor de nivel en cada tanque genera una tensión a nivel ALTO cuando el nivel de líquido en el tanque cae por debajo de un punto especificado.

Diseñar un circuito para supervisar el nivel del elemento químico en cada tanque, que indique cuándo el nivel de dos tanques cualesquiera cae por debajo del punto especificado.

Solución El circuito AND-OR de la Figura 5.2 tiene entradas procedentes de los sensores de los tanques *A*, *B* y *C*. La puerta AND G_1 comprueba el nivel en los

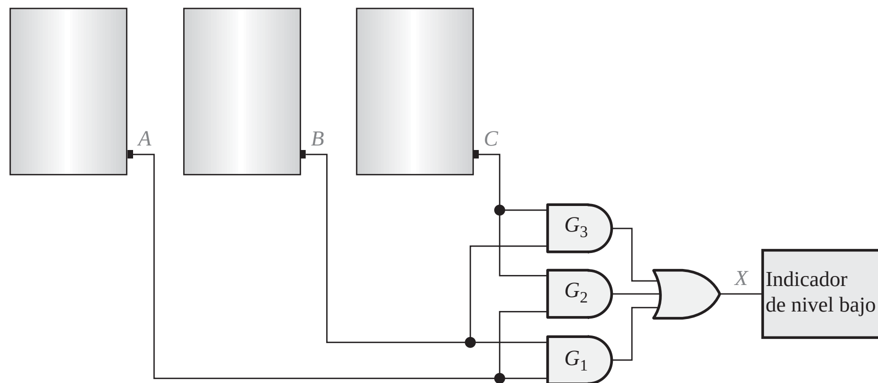


FIGURA 5.2

tanques A y B, la puerta G_2 comprueba los tanques A y C, y la puerta G_3 comprueba los tanques B y C. Cuando el nivel del elemento químico en dos tanques cualesquiera desciende demasiado, una de las puertas AND tendrá a nivel ALTO ambas entradas, haciendo que su salida sea un nivel ALTO, por lo que la salida final X de la puerta OR estará a nivel ALTO. Esta salida a nivel ALTO se usa entonces para activar un indicador, tal como una alarma luminosa o audible, como muestra la figura.

Problema relacionado* Escribir la expresión booleana en forma de suma de productos para el circuito lógico AND-OR de la Figura 5.2.

* Las respuestas se encuentran al final del capítulo.

El diagrama lógico de la Figura 5.3(a) muestra un circuito AND-OR-Inversor y el desarrollo de la expresión de salida como producto de sumas. En la parte (b) de la figura se presenta el símbolo rectangular estándar ANSI. En general, un circuito AND-OR-Inversor puede tener cualquier número de puertas AND, y cada una de ellas puede tener cualquier número de entradas.

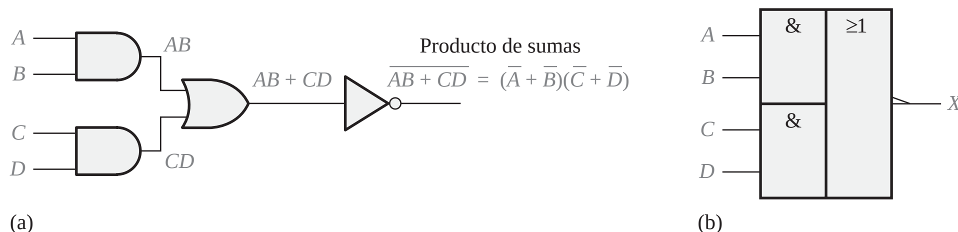


FIGURA 5.3 Un circuito AND-OR-Inversor genera una salida que es un producto de sumas.

La operación lógica del circuito AND-OR-Inversor de la Figura 5.3 se define como:

En un circuito lógico AND-OR-Inversor de 4 entradas, la entrada *X* es un nivel BAJO (0) si las dos entradas *A* y *B* están a nivel ALTO (1), o si las dos entradas *C* y *D* están a nivel ALTO (1).

Se puede desarrollar la tabla de verdad a partir de la Tabla 5.1, tabla de verdad del circuito AND-OR, cambiando simplemente, en la columna de salida, todos los 1s por 0s y todos los 0s por 1s.

EJEMPLO 5.2

Los sensores colocados en los tanques químicos del Ejemplo 5.1 se reemplazan por un nuevo modelo que genera una tensión a nivel BAJO en lugar de una tensión a nivel ALTO cuando el nivel de líquido en el tanque cae por debajo del punto crítico.

Modificar el circuito de la Figura 5.2 para trabajar con los diferentes niveles de entrada y generar una salida a nivel ALTO que active el indicador cuando el nivel de dos tanques caiga por debajo del punto crítico. Realizar el diagrama lógico.

Solución Los sensores de los tanques *A, B* y *C* se conectan a las entradas del circuito AND-OR-Inversor, como se muestra en la Figura 5.4. La puerta AND G_1 comprueba el nivel en los tanques A y B, la puerta G_2 comprueba los tanques A y C, y la puerta G_3 comprueba los tanques B y C. Cuando el nivel del elemento

químico en dos tanques cualesquiera desciende, al menos una de las entradas de cada una de las puertas AND estará a nivel BAJO, haciendo que su salida sea un nivel BAJO, por lo que la salida final X del inversor estará a nivel ALTO. Esta salida a nivel ALTO se usa entonces para activar un indicador.

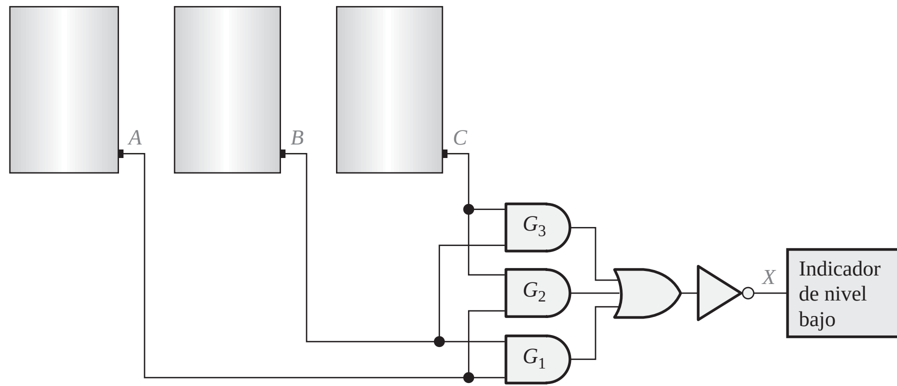
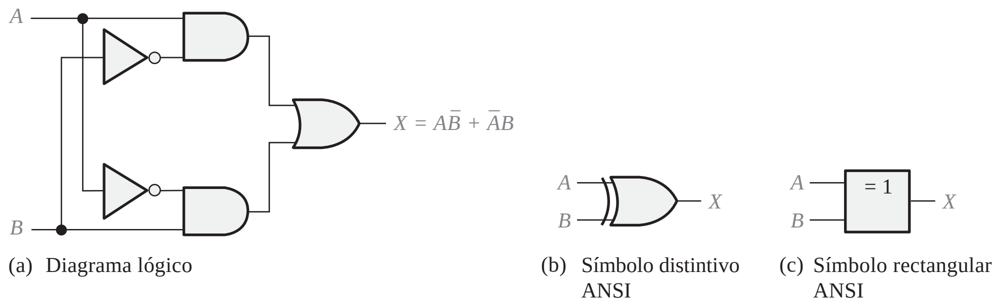


FIGURA 5.4

Problema relacionado Escribir la expresión booleana para el circuito lógico AND-OR-Inversor de la Figura 5.4 y mostrar que la salida es un nivel ALTO (1) cuando dos entradas cualesquiera de entre A, B y C estén a nivel BAJO (0).

Circuito lógico OR-exclusiva

En el Capítulo 3 se ha presentado la puerta OR-exclusiva. Aunque, debido a su importancia este circuito se considera como una puerta lógica con su propio símbolo distintivo, realmente es una combinación de dos puertas AND, una puerta OR y dos inversores, tal y como muestra la Figura 5.5(a). En las Figuras 5.5 (b) y (c) se presentan los dos símbolos lógicos estándar ANSI.



(a) Diagrama lógico (b) Símbolo distintivo ANSI (c) Símbolo rectangular ANSI

FIGURA 5.5 Diagrama lógico y símbolos del circuito OR-exclusiva.

La expresión de salida para el circuito de la Figura 5.5 es

$$X = A\overline{B} + \overline{A}B$$

La evaluación de esta expresión se muestra en la tabla de verdad de la Tabla 5.2. Observe que la salida está a nivel ALTO sólo cuando las dos entradas están a niveles opuestos. A menudo, se emplea el operador especial OR-exclusiva ⊕, por lo que la expresión $X = A\overline{B} + \overline{A}B$ puede enunciarse como "*X* es igual a *A* OR-Exclusiva *B*" y puede expresarse como:

$$X = A \oplus B$$

A	*B*	*X*
0	0	0
0	1	1
1	0	1
1	1	0

TABLA 5.2 Tabla de verdad para la puerta OR-exclusiva.

Circuito lógico NOR-exclusiva

Como sabemos, el complemento de la función OR-exclusiva es la función NOR-exclusiva, la cual se obtiene del siguiente modo:

$$X = \overline{A\overline{B} + \overline{A}B} = \overline{(A\overline{B})}\,\overline{(\overline{A}B)} = (\overline{A} + B)(A + \overline{B}) = \overline{A}\,\overline{B} + AB$$

Observe que la salida *X* es un nivel ALTO sólo cuando las dos entradas, *A* y *B*, están al mismo nivel.

La función NOR-exclusiva puede implementarse invirtiendo la salida de un circuito OR-exclusiva, como muestra la Figura 5.6(a), o bien se puede implementar directamente a partir de la expresión $\overline{A}\,\overline{B} + AB$, como muestra la parte (b) de la figura.

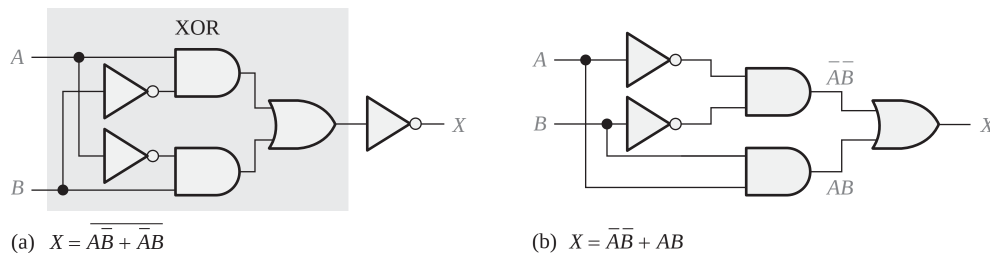


FIGURA 5.6 Dos formas equivalentes de implementar el circuito NOR-exclusiva.

REVISIÓN DE LA SECCIÓN 5.1

Las respuestas se encuentran al final del capítulo.

1. Determinar la salida (1 o 0) de un circuito AND-OR-Inversor de 4 variables para cada una de las siguientes condiciones de entrada:
   (a) $A = 1, B = 0, C = 1, D = 0$
   (b) $A = 1, B = 1, C = 0, D = 1$
   (c) $A = 0, B = 1, C = 1, D = 1$
2. Determinar la salida (1 o 0) de una puerta OR-exclusiva para cada una de las siguientes condiciones de entrada:

(a) $A = 1, B = 0$ **(b)** $A = 1, B = 1$

(c) $A = 0, B = 1,$ **(d)** $A = 0, B = 0$

3. Desarrollar la tabla de verdad para un determinado circuito lógico de 3 entradas con la siguiente expresión de salida $X = A\overline{B}C + \overline{A}BC + \overline{A}\overline{B}\overline{C} + AB\overline{C} + ABC$.
4. Dibujar un diagrama lógico para un circuito NOR-exclusiva.

5.2 IMPLEMENTACIÓN DE LA LÓGICA COMBINACIONAL

En esta sección se emplean ejemplos para ilustrar cómo implementar un circuito lógico a partir de una expresión booleana o una tabla de verdad. También se tratará la minimización de un circuito lógico utilizando los métodos vistos en el Capítulo 4.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Implementar un circuito lógico a partir de una expresión booleana. ■ Implementar un circuito lógico a partir de una tabla de verdad. ■ Minimizar un circuito lógico.

Obtención del circuito lógico a partir de una expresión booleana

▲ *Para toda expresión booleana existe un circuito lógico y para todo circuito lógico existe una expresión booleana.*

Examinemos la siguiente expresión booleana:

$$X = AB + CDE$$

Una rápida inspección revela que esta expresión está formada por dos términos, *AB* y *CDE*, y tienen un dominio de cinco variables. El primer término está definido por la operación AND entre *A* y *B*, y el segundo término queda definido por la multiplicación (AND) de *C*, *D* y *E*. Después, los dos términos se suman (operación OR) para dar lugar a la salida *X*. Estas operaciones se indican en la siguiente estructura de la expresión:

AND

$$X = AB + CDE$$

OR

Observe que, en esta expresión, las operaciones AND son dos términos individuales, *AB* y *CDE*, que deben efectuarse *antes* de aplicar la operación OR.

Para implementar esta expresión booleana se requiere una puerta AND de 2 entradas para obtener el término *AB*, y una puerta AND de tres entradas para generar el término *CDE*. Para combinar los dos términos obtenidos en las puertas AND se requiere una puerta OR de 2 entradas. El resultado se muestra en la Figura 5.7.

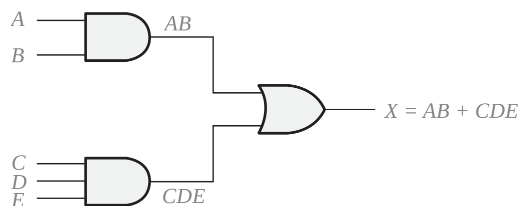
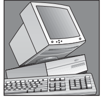


FIGURA 5.7 Circuito lógico para $X = AB + CDE$.



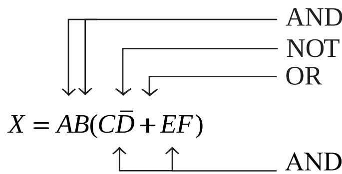
NOTAS INFORMÁTICAS

Muchos programas de control requieren que una computadora realice operaciones lógicas. Un controlador es un programa de control que se emplea con los periféricos de la computadora. Por ejemplo, un controlador de ratón requiere pruebas lógicas para determinar si se ha pulsado un botón y más operaciones lógicas para determinar si se ha movido en sentido horizontal o en sentido vertical. En el núcleo de un procesador se encuentra la unidad aritmético lógica (ALU), que realiza dichas operaciones lógicas mediante instrucciones de programa. Toda la lógica que se describe en este capítulo también puede ser realizada por la ALU, proporcionando las instrucciones adecuadas.

Veamos otro ejemplo, vamos a implementar la siguiente expresión:

$$X = AB(C\bar{D} + EF)$$

Un examen de esta expresión muestra que se aplica la operación AND a los términos AB y $(C\bar{D} + EF)$. El término $C\bar{D} + EF$ se obtiene aplicando primero la operación AND a C y $\bar{D}$ y luego a E y F, y aplicando por último la operación OR a estos dos términos. Esta estructura se indica respecto de la expresión del siguiente modo:



Antes de poder implementar la expresión completa, hay que crear el término suma $C\bar{D} + EF$. Para ello, previamente hay que disponer de los términos $C\bar{D}$ y EF, pero antes de obtener el término $C\bar{D}$ es necesario tener $\bar{D}$. Luego, como puede ver, las operaciones lógicas deben efectuarse en el orden adecuado.

Las puertas lógicas necesarias para implementar $X = AB(C\bar{D} + EF)$ son las siguientes:

1. Un inversor para obtener $\bar{D}$.
2. Dos puertas AND de 2 entradas para obtener $C\bar{D}$ y EF.
3. Una puerta OR de 2 entradas para obtener $C\bar{D} + EF$.
4. Una puerta AND de 3 entradas para generar X.

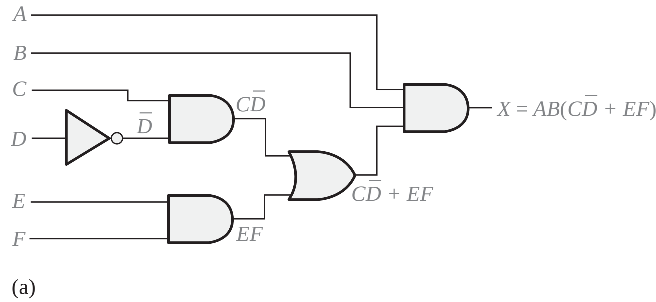
En la Figura 5.8(a) se muestra el circuito lógico correspondiente a esta expresión. Observe que hay un máximo de tres puertas y un inversor entre una entrada y la salida del circuito (de la entrada D a la salida). A menudo, el retardo de propagación a través del circuito lógico es una consideración de gran importancia. Los retardos de propagación se suman, luego cuantos más inversores y puertas haya entre la entrada y la salida, mayor será el retardo de propagación.

A menos que un término intermedio, tal como $C\bar{D} + EF$ de la Figura 5.8(a), se requiera como salida para algún otro propósito, usualmente lo mejor es reducir el circuito a su suma de productos con el fin de reducir el tiempo de retardo de propagación total. La expresión se convierte en suma de productos como sigue, y el circuito resultante se muestra en la Figura 5.8(b).

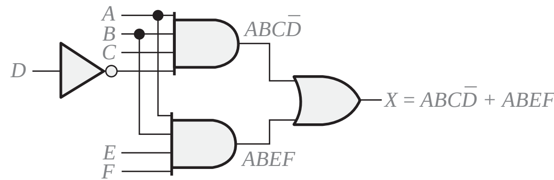
$$AB(C\bar{D} + EF) = ABC\bar{D} + ABEF$$

Obtención del circuito lógico a partir de la tabla de verdad

Si en lugar de partir de una expresión se parte de una tabla de verdad, puede escribirse la suma de productos que se obtiene de la tabla de verdad, y luego implementar el circuito lógico. La Tabla 5.3 especifica una función lógica.



(a)



(b) Implementación de la suma de productos del circuito de la parte (a)

FIGURA 5.8 Circuitos lógicos para la expresión $X = AB(C\overline{D}+EF) = ABC\overline{D} + ABEF$.

Entradas			Salida	
A	***B***	***C***	***X***	**Término producto**
0	0	0	0	
0	0	1	0	
0	1	0	0	
0	1	1	1	$\overline{A}BC$
1	0	0	1	$A\overline{B}\overline{C}$
1	0	1	0	
1	1	0	0	
1	1	1	0	

TABLA 5.3

La expresión booleana suma de productos que se obtiene a partir de la tabla de verdad, haciendo la suma (OR) de los productos para los que $X = 1$ es

$$X = \overline{A}BC + A\overline{B}\overline{C}$$

El primer término de la expresión se obtiene calculando el producto lógico (AND) de las tres variables $\overline{A}, B$ y C. El segundo término se obtiene multiplicando (operación AND) las tres variables A, $\overline{B}$ y $\overline{C}$.

Las puertas lógicas necesarias para implementar esta expresión son: tres inversores para obtener las variables $\overline{A}, \overline{B}$ y $\overline{C}$; dos puertas AND de 3 entradas para obtener los términos $\overline{A}BC$ y $A\overline{B}\overline{C}$ y una puerta OR de 2 entradas para obtener la forma de la función final de salida $\overline{A}BC + A\overline{B}\overline{C}$.

La Figura 5.9 ilustra la implementación de esta función lógica.

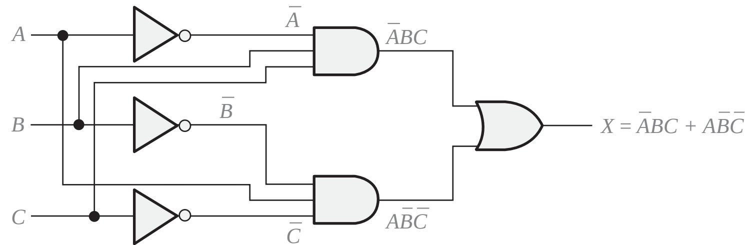


FIGURA 5.9 Circuito lógico para $X = \bar{A}BC + A\bar{B}\bar{C}$.

EJEMPLO 5.3

Diseñar un circuito lógico para implementar la operación especificada en la tabla de verdad indicada en la Tabla 5.4.

Entradas			Salida	
A	*B*	*C*	*X*	Término producto
0	0	0	0	
0	0	1	0	
0	1	0	0	
0	1	1	1	$\bar{A}BC$
1	0	0	0	
1	0	1	1	$A\bar{B}C$
1	1	0	1	$AB\bar{C}$
1	1	1	0	

TABLA 5.4

Solución Observe que $X = 1$ sólo para tres de las condiciones de entrada. Por tanto, la expresión lógica es: $X = \bar{A}BC + A\bar{B}C + AB\bar{C}$.

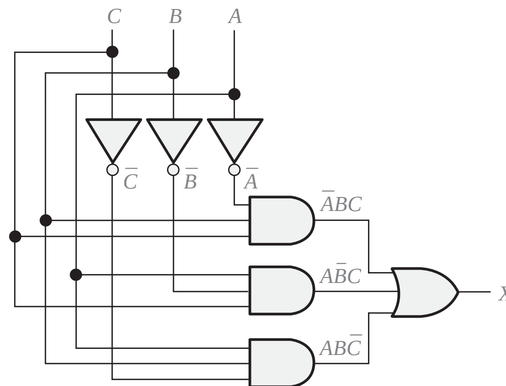


FIGURA 5.10

Las puertas lógicas requeridas son tres inversores, tres puertas AND de 3 entradas y una puerta OR de 3 entradas. El circuito lógico es el que se muestra en la Figura 5.10.

Problema relacionado Determinar si el circuito lógico de la Figura 5.10 puede simplificarse.

EJEMPLO 5.4

Desarrollar un circuito lógico con cuatro variables de entrada que sólo genera un 1 en la salida cuando tres variables de entrada son 1.

Solución Para cuatro variables, existen dieciséis posibles combinaciones; de éstas las que contienen tres 1s son las que se enumeran en la Tabla 5.5, junto con el correspondiente término producto.

A	*B*	*C*	*D*	Término producto
0	1	1	1	$\overline{A}BCD$
1	0	1	1	$A\overline{B}CD$
1	1	0	1	$AB\overline{C}D$
1	1	1	0	$ABC\overline{D}$

TABLA 5.5

Se aplica la operación OR a los productos para obtener la siguiente expresión:

$$X = \overline{A}BCD + A\overline{B}CD + AB\overline{C}D + ABC\overline{D}$$

Esta expresión se implementa con el circuito lógico AND-OR de la Figura 5.11.

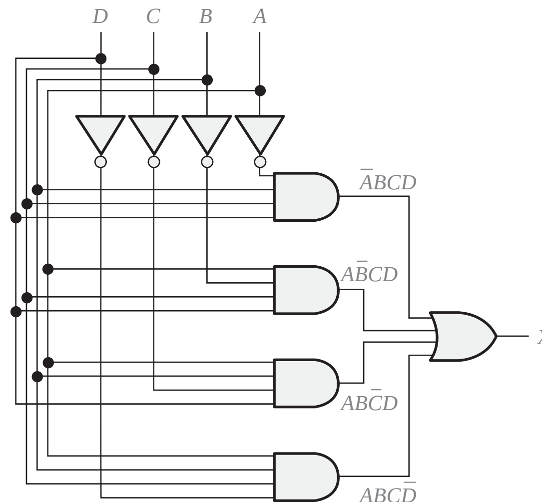


FIGURA 5.11

Problema relacionado Determinar si el circuito lógico de la Figura 5.11 puede simplificarse.

EJEMPLO 5.5

Reducir el circuito lógico combinacional de la Figura 5.12 a una forma mínima.

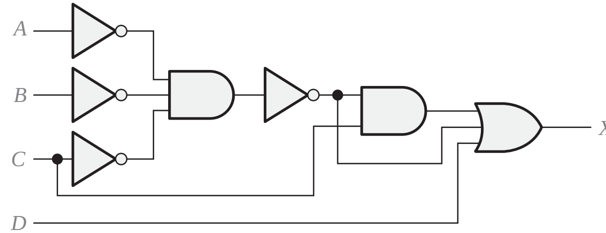


FIGURA 5.12

Solución La expresión para la salida del circuito es:

$$X = (\overline{\overline{A}\overline{B}\overline{C}})C + \overline{\overline{A}\overline{B}\overline{C}} + D$$

Aplicando el teorema de DeMorgan y el álgebra booleana se tiene

$$\begin{aligned} X &= (\overline{\overline{A}} + \overline{\overline{B}} + \overline{\overline{C}})C + \overline{\overline{A}} + \overline{\overline{B}} + \overline{\overline{C}} + D \\ &= AC + BC + CC + A + B + C + D \\ &= AC + BC + C + A + B + \not{C} + D \\ &= C(A + B + 1) + A + B + D \\ X &= A + B + C + D \end{aligned}$$

El circuito simplificado es una puerta OR de cuatro entradas, como muestra la Figura 5.13.

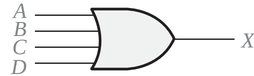


FIGURA 5.13

Problema relacionado Verificar la expresión minimizada $A + B + C + D$ utilizando un mapa de Karnaugh.

EJEMPLO 5.6

Minimizar el circuito lógico combinacional de la Figura 5.14. Los inversores para las variables complementadas no se muestran.

Solución La expresión de salida es:

$$X = A\overline{B}\overline{C} + AB\overline{C}\overline{D} + \overline{A}\overline{B}\overline{C}D + \overline{A}\overline{B}\overline{C}\overline{D}$$

Desarrollando el primer término para incluir las variables que faltan D y $\overline{D}$ tenemos,

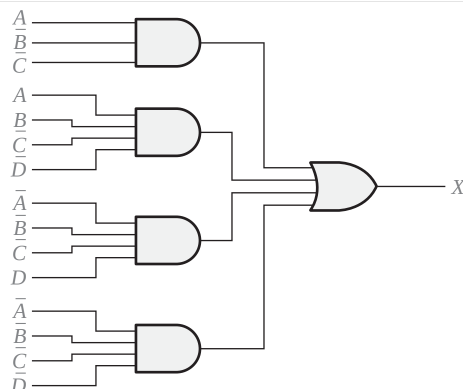
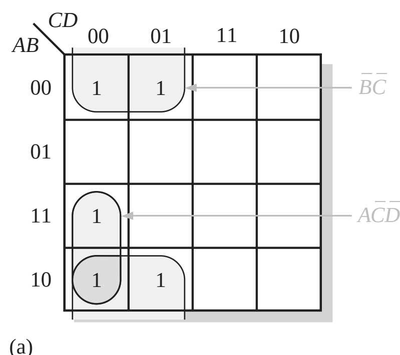


FIGURA 5.14

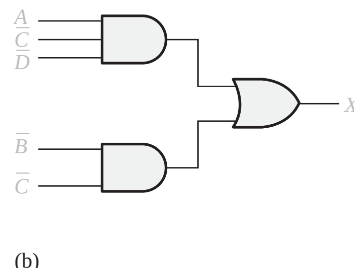
$$X = A\overline{B}\,\overline{C}(D+\overline{D}) + AB\overline{C}\,\overline{D} + \overline{A}\,\overline{B}\,\overline{C}D + \overline{A}\,\overline{B}\,\overline{C}\,\overline{D}$$

$$= A\overline{B}\,\overline{C}D + A\overline{B}\,\overline{C}\,\overline{D} + AB\overline{C}\,\overline{D} + \overline{A}\,\overline{B}\,\overline{C}D + \overline{A}\,\overline{B}\,\overline{C}\,\overline{D}$$

Esta suma de productos completa se traslada y simplifica en el mapa de Karnaugh de la Figura 5.15(a). La implementación simplificada se muestra en la parte (b). Los inversores no se muestran.



(a)



(b)

FIGURA 5.15

Problema relacionado Desarrollar el producto de sumas equivalente del circuito de la Figura 5.15(b).

REVISIÓN DE LA SECCIÓN 5.2

1. Implementar las siguientes expresiones booleanas tal y como se definen.
   (a) $X = ABC + AB + AC$ **(b)** $X = AB(C + DE)$
2. Desarrollar un circuito lógico que genere un 1 en su salida cuando sus tres entradas están a 1 o cuando sus tres entradas están a 0.
3. Simplificar los circuitos de la cuestión 1 utilizando la suma de productos mínima.

5.3 LA PROPIEDAD UNIVERSAL DE LAS PUERTAS NAND Y NOR

Hasta este momento se han estudiado los circuitos combinacionales que se implementan con puertas AND, puertas OR e inversores. En esta sección, se va a tratar la propiedad universal de la puerta NAND y de la puerta NOR. La universalidad de la puerta NAND significa que puede utilizarse como un inversor y que pueden emplearse combinaciones de la puerta NAND para implementar las operaciones AND, OR y NOR. Del mismo modo, la puerta NOR se puede utilizar para implementar el inversor (NOT) y las operaciones AND, OR y NAND.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Utilizar las puertas NAND para implementar el inversor, la puerta AND, la puerta OR y la puerta NOR. ■ Utilizar las puertas NOR para implementar el inversor, la puerta AND, la puerta OR y la puerta NAND.

La puerta NAND como elemento lógico universal

▲ *Las puertas NAND pueden emplearse para generar cualquier función lógica.*

La puerta NAND es una ***puerta universal*** porque puede utilizarse para generar las funciones NOT, AND, OR y NOR. Se puede obtener un inversor a partir de una puerta NAND conectando juntas todas las entradas, dando lugar a una única entrada, como se muestra en la Figura 5.16(a) con una puerta de 2 entradas. La operación

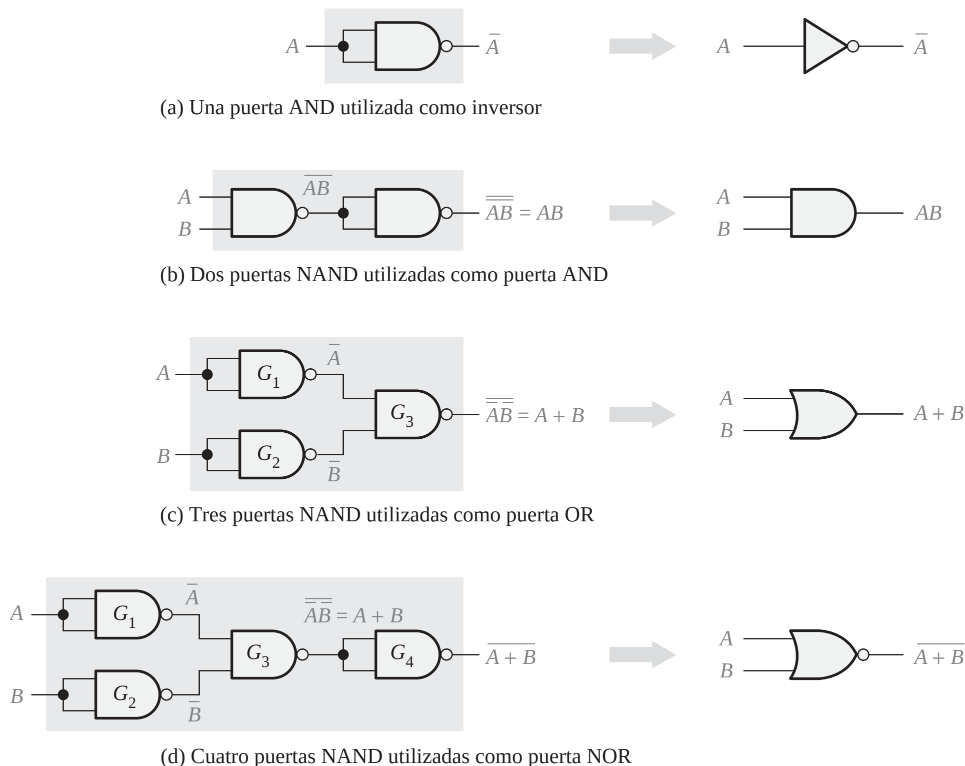


FIGURA 5.16 Aplicación universal de las puertas NAND.

AND se puede generar utilizando sólo puertas NAND, como muestra la Figura 5.16(b). La operación OR se puede obtener con varias puertas NAND, como ilustra la parte (c). Por último, la operación NOR se obtiene como se indica en la parte (d) de la figura.

En la Figura 5.16(b), se utiliza una puerta NAND para invertir (complementar) la salida de una puerta NAND para obtener una función AND, como indica la siguiente ecuación:

$$X = \overline{\overline{AB}} = AB$$

En la Figura 5.16(c), las puertas NAND G_1 y G_2 se emplean para invertir las dos variables de entrada antes de aplicarlas a la puerta NAND G_3. La salida final de la puerta OR se obtiene aplicando el teorema de DeMorgan del siguiente modo:

$$X = \overline{\overline{A}\,\overline{B}} = A + B$$

En la Figura 5.16(d), la puerta NAND G_4 se utiliza como un inversor conectado al circuito de la parte (c) con el fin de obtener la operación NOR $A + B$.

La puerta NOR como un elemento lógico universal

Al igual que la puerta NAND, la puerta NOR se puede utilizar para generar las funciones NOT, AND, OR y NAND. Un circuito NOT, o inversor, puede obtenerse a partir de una puerta NOR conectando todas sus

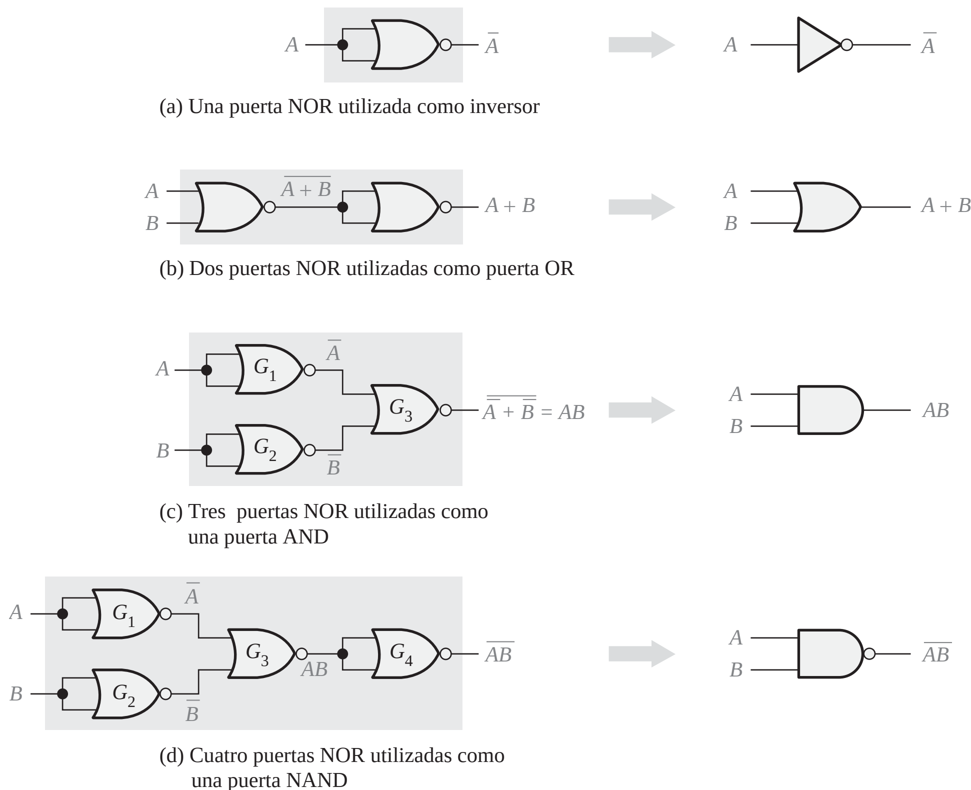


FIGURA 5.17 Aplicación universal de las puertas NOR.

▲ *Las puertas NOR pueden emplearse para generar cualquier función lógica.*

entradas juntas para tener una única puerta, como se muestra en la Figura 5.17(a) con una puerta de 2 entradas. También puede obtenerse una puerta OR a partir de puertas NOR, como se ilustra en la Figura 5.17(b). Un puerta AND puede construirse utilizando puertas NOR como muestra la Figura 5.17(c). En este caso, las puertas NOR G_1 y G_2 se usan como inversores y la salida final se obtiene aplicando el teorema de DeMorgan del siguiente modo:

$$X = \overline{\overline{A} + \overline{B}} = AB$$

La Figura 5.17(d) muestra cómo se usan las puertas NOR para obtener una función NAND.

REVISIÓN DE LA SECCIÓN 5.3

1. Utilizando puertas NAND implementar las siguientes expresiones:
   (a) $X = \overline{A} + B$ **(b)** $X = A\overline{B}$
2. Utilizando puertas NOR implementar las siguientes expresiones:
   (a) $X = \overline{A} + B$ **(b)** $X = A\overline{B}$

5.4 LÓGICA COMBINACIONAL CON PUERTAS NAND Y NOR

En esta sección se verá cómo se usan las puertas NAND y NOR para implementar una función lógica. Recuerde del Capítulo 3 que la puerta NAND tiene una operación equivalente denominada negativa-OR, y que la puerta NOR tiene una operación equivalente denominada negativa-AND. Veremos cómo el uso de los símbolos adecuados para representar las operaciones equivalentes hace la “lectura” del diagrama lógico más fácil.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Utilizar puertas NAND para implementar una función lógica. ■ Utilizar puertas NOR para implementar una función lógica. ■ Utilizar el símbolo apropiado en un diagrama lógico.

Circuito lógico NAND

Como ya se ha dicho, una puerta NAND puede expresarse como una función NAND o una función negativa-OR, ya que por el teorema de DeMorgan:

$$\overline{AB} = \overline{A} + \overline{B}$$

NAND ⟶ ↑ ↑ ⟵ negativa-OR

Considerando el circuito lógico de la Figura 5.18, la expresión de salida se desarrolla según los pasos siguientes:

$$\begin{aligned} X &= \overline{(\overline{AB})(\overline{CD})} \\ &= \overline{(\overline{A} + \overline{B})(\overline{C} + \overline{D})} \\ &= \overline{(\overline{A} + \overline{B})} + \overline{(\overline{C} + \overline{D})} \\ &= \overline{\overline{A}}\,\overline{\overline{B}} + \overline{\overline{C}}\,\overline{\overline{D}} \\ &= AB + CD \end{aligned}$$

Como puede ver en la Figura 5.18, la expresión de salida, $AB + CD$, corresponde a la forma de dos términos que se multiplican (AND) y luego se suman (OR). Esta expresión muestra que las puertas G_2 y G_3 actúan

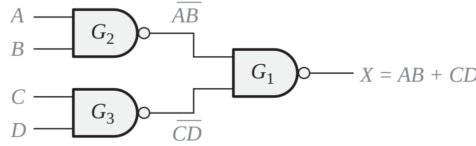
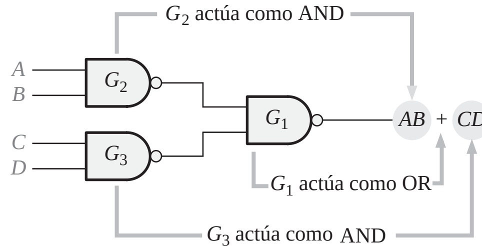
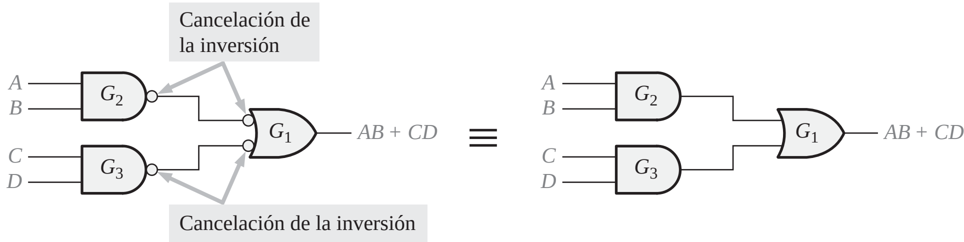


FIGURA 5.18 Circuito lógico NAND para $X = AB + CD$.

como puertas AND, y la puerta G_1 actúa como puerta OR, como ilustra la Figura 5.19(a). En la parte (b) de esta figura se presenta este circuito con los símbolos NAND para las puertas G_2 y G_3, y un símbolo de la puerta negativa-OR para la puerta G_1.



(a) Diagrama lógico NAND original que muestra la operación de la puerta correspondiente a la expresión de salida.



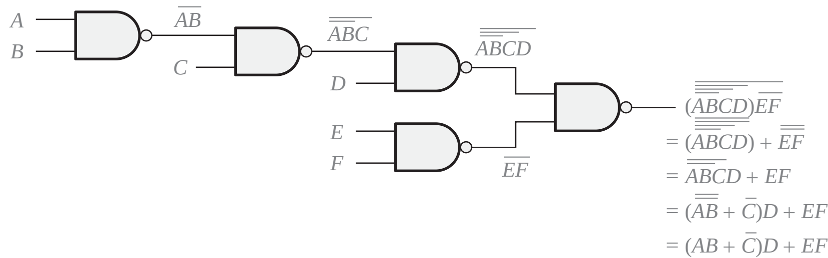
(b) Diagrama lógico equivalente NAND/Negativa-OR. (c) Equivalente AND-OR.

FIGURA 5.19 Desarrollo del equivalente AND-OR del circuito de la Figura 5.18.

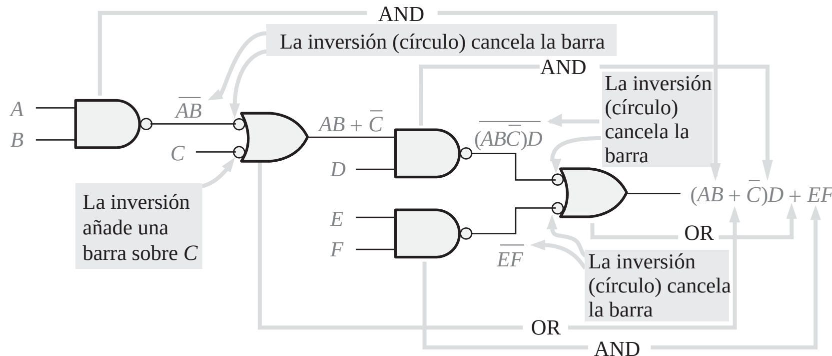
Fíjese en la Figura 5.19(b) en las conexiones círculo-círculo entre las salidas de las puertas G_2 y G_3, y las entradas de G_1. *Puesto que un círculo indica una inversión, dos círculos conectados representan una doble inversión y, por tanto, se cancelan entre sí*. Esta cancelación de inversión se ha podido ver en el desarrollo anterior para la expresión de salida $AB + CD$, y se indica por la ausencia, en la misma, de términos con una barra encima. Luego el circuito de la Figura 5.19(b) es *efectivamente* un circuito AND-OR, como se muestra en la Figura 5.19(c).

Diagrama lógico NAND utilizando símbolos duales. Todos los diagramas lógicos que utilizan puertas NAND deberían dibujarse utilizando el símbolo NAND o el símbolo equivalente negativa-OR para representar cada puerta, con el fin de reflejar la operación de la puerta dentro del circuito lógico. Los símbolos NAND y negativa-OR se denominan símbolos duales. Cuando se dibuja un diagrama lógico NAND, siempre se empleen los símbolos de puerta de tal forma que cada una de las conexiones entre la salida de una puerta y la entrada de otra sea una conexión círculo-círculo o una conexión no círculo-no círculo.

La Figura 5.20 ilustra el procedimiento de utilización de los símbolos duales adecuados para un circuito NAND con varios niveles de puertas. Aunque es correcto utilizar siempre símbolos NAND, como muestra la



(a) Se necesitan varios pasos del álgebra booleana para llegar a la expresión de salida final.



(b) La expresión de salida puede obtenerse directamente a partir de la función del símbolo de cada puerta del diagrama.

FIGURA 5.20 Ejemplo de utilización de los símbolos duales apropiados en un diagrama lógico NAND.

Figura 5.20(a), el diagrama de la parte (b) es más fácil de "leer" y es preferible. Como puede verse en la Figura 5.20(b), la puerta de salida se ha representado con un símbolo negativa-OR. El símbolo NAND se emplea para los niveles de puertas anteriores a la puerta de salida y los símbolos para los sucesivos niveles de puertas se alternan según se alejan de la salida.

La forma de las puertas indica cómo aparecerán sus entradas en la ecuación de salida y, por tanto, cómo funciona la puerta dentro del circuito lógico. Cuando se usa el símbolo NAND, las entradas aparecen multiplicadas (AND) en la expresión de salida, y cuando se usa el símbolo negativa-OR las entradas aparecen sumadas (OR), como ilustra la Figura 5.20(b). Puede ver que en el diagrama de símbolos duales de la parte (b) de la figura es mucho más fácil determinar directamente la expresión de salida, ya que cada símbolo de puerta indica las relaciones de sus variables de entrada, tal y como aparecen en la expresión de salida.

EJEMPLO 5.7

Volver a dibujar el diagrama lógico y desarrollar la expresión de salida para el circuito de la Figura 5.21, utilizando los símbolos duales adecuados.

Solución Dibujamos de nuevo el diagrama lógico de la Figura 5.21 utilizando símbolos equivalentes negativa-OR como se muestra en la Figura 5.22. La expresión de *X* obtenida directamente de la operación lógica que indica cada puerta es:

$$X = (\overline{A} + \overline{B})C + (\overline{D} + \overline{E})F$$

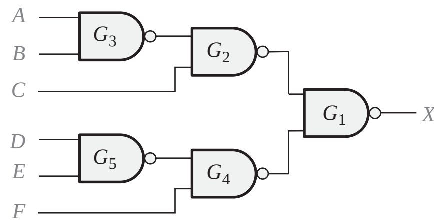


FIGURA 5.21

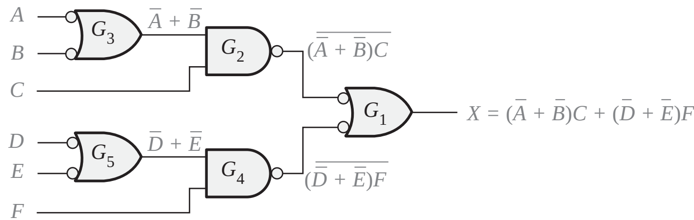


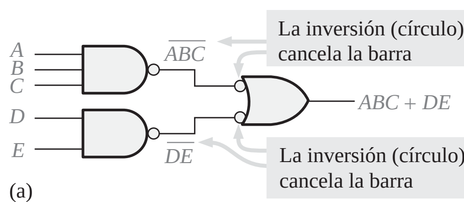
FIGURA 5.22

Problema relacionado Obtener la expresión de salida a partir de la Figura 5.21, y demostrar que es equivalente a la expresión obtenida como solución.

EJEMPLO 5.8

Implementar las siguientes expresiones mediante lógica NAND usando los símbolos duales apropiados:
(a) $ABC + DE$ **(b)** $ABC + \bar{D} + \bar{E}$

Solución *Véase* la Figura 5.23.



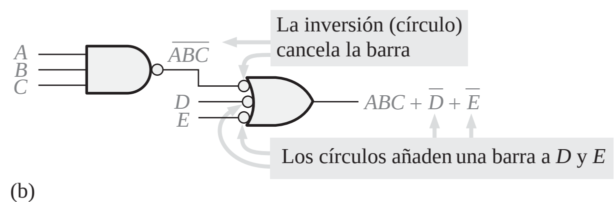


FIGURA 5.23

Problema relacionado Convertir los circuitos NAND de las Figuras 5.23(a) y (b) a su equivalente lógico AND-OR.

Lógica NOR

Una puerta NOR puede funcionar como NOR o como ***negativa-AND***, como demuestra el teorema de DeMorgan:

$$\overline{A+B} = \bar{A}\bar{B}$$

NOR ↑ (en $\overline{A+B}$) ↑ negativa-AND (en $\bar{A}\bar{B}$)

Consideremos el diagrama lógico NOR de la Figura 5.24. La expresión de salida se desarrolla así:

$$X = \overline{\overline{A+B}+\overline{C+D}} = (\overline{\overline{A+B}})(\overline{\overline{C+D}}) = (A+B)(C+D)$$

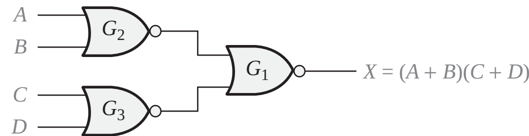


FIGURA 5.24 Diagrama lógico NOR para $X = (A+B)\ (C + D)$.

Como puede verse en la Figura 5.24, la expresión de salida $(A + B)(C + D)$ está formada por dos términos a los que primero se les aplica la operación OR y luego la operación AND. Esto implica que las puertas G_2 y G_3 operan como puertas OR, y la puerta G_1 como puerta AND, como muestra la Figura 5.25(a). Este circuito se ha dibujado de nuevo en la parte (b) de la figura con un símbolo negativa-AND para la puerta G_1.

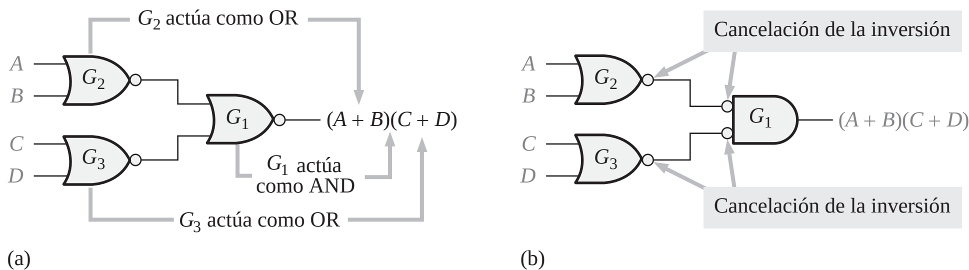
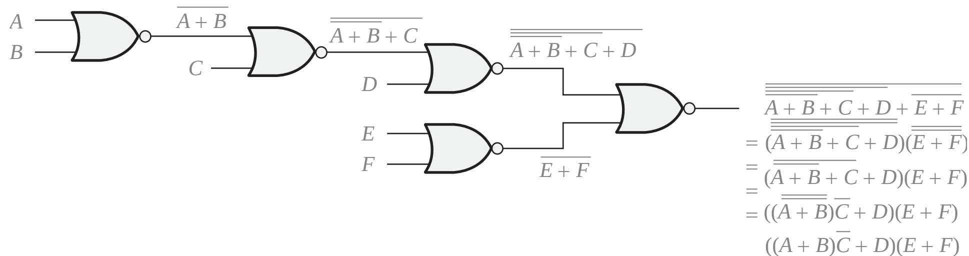


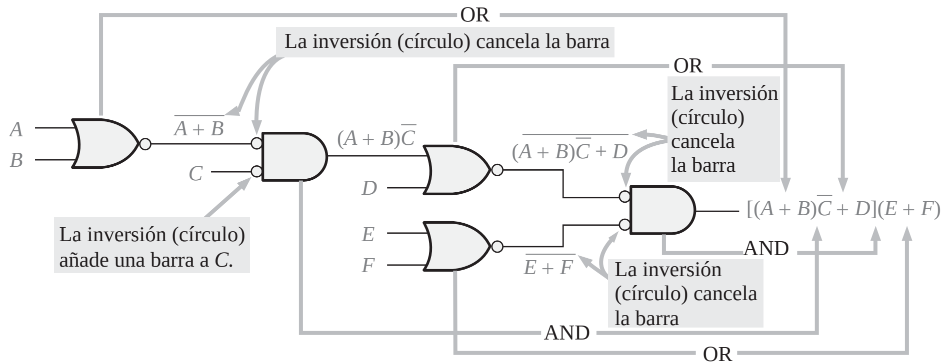
FIGURA 5.25

Diagrama lógico NOR utilizando símbolos duales. Como en el diagrama lógico NAND, el propósito de utilizar los símbolos duales es hacer más fácil la lectura y el análisis del diagrama, lo que se ilustra en el circuito lógico NOR de la Figura 5.26. Cuando el circuito de la parte (a) se redibuja con símbolos duales, se obtiene el circuito de la parte (b) de la figura; observe que todas las conexiones de salida-entrada son círculo-círculo o no círculo-no círculo. De nuevo, puede comprobar que la forma de cada puerta indica el tipo de término (AND u OR) que producirá en la expresión de salida, lo que hace que sea más fácil determinar la expresión de salida y más fácil también analizar el diagrama lógico.



(a) La expresión de salida final se obtiene después de aplicar varios pasos del álgebra booleana.

FIGURA 5.26 Ilustración del uso de los símbolos duales apropiados en un diagrama lógico NOR. (*Continúa*)



(b) La expresión de salida puede obtenerse directamente de la función de cada símbolo de puerta del diagrama.

FIGURA 5.26 (*Continuación*).

EJEMPLO 5.9

Utilizando los símbolos duales apropiados, dibujar de nuevo el diagrama lógico y desarrollar la expresión de salida para el circuito de la Figura 5.27.

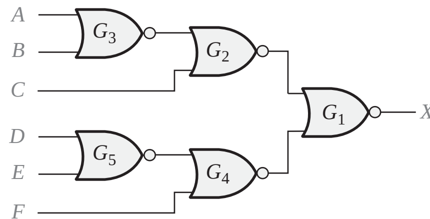


FIGURA 5.27

Solución En la Figura 5.28 se muestra el nuevo diagrama lógico utilizando el símbolo de la puerta negativa-AND equivalente. La expresión X se obtiene directamente a partir de la función lógica de cada puerta.

$$X = (\overline{A}\overline{B} + C)(\overline{D}\overline{E} + F)$$

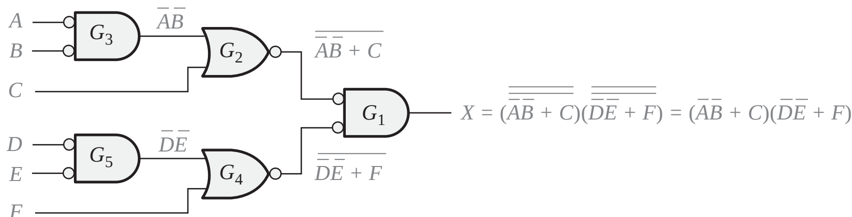


FIGURA 5.28

Problema relacionado Demostrar que la salida del circuito NOR de la Figura 5.27 es igual que la del circuito de la Figura 5.28.

REVISIÓN DE LA SECCIÓN 5.4

1. Implementar la expresión $X = \overline{(\overline{A} + \overline{B} + \overline{C})DE}$ usando un diagrama lógico NAND.
2. Implementar la expresión $X = \overline{\overline{A}\overline{B}\overline{C} + (D + E)}$ usando un diagrama lógico NOR.

5.5 FUNCIONAMIENTO DE LOS CIRCUITOS LÓGICOS CON TRENES DE IMPULSOS

En esta sección se examinan varios ejemplos de circuitos lógicos combinacionales con entradas que son trenes de impulsos. Hay que tener en mente que la operación lógica de las puertas es la misma para impulsos que para niveles continuos de entrada. En cualquier instante determinado, la salida de un circuito lógico depende de sus entradas en ese instante, por lo que son de importancia capital las variaciones con el tiempo de las entradas.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Analizar cualquier circuito lógico combinacional con trenes de impulsos como entradas. ■ Desarrollar el diagrama de tiempos para cualquier circuito lógico combinacional de acuerdo con las entradas especificadas.

La operación lógica que realiza una puerta es la misma independientemente de que a sus entradas se apliquen impulsos o niveles constantes. La naturaleza de las entradas (impulsos o niveles constantes) no altera la tabla de verdad de un circuito. Los ejemplos de esta sección ilustran el análisis de circuitos lógicos combinacionales con impulsos de entrada.

A continuación, se presenta un repaso de la operación lógica que realiza cada puerta, con el fin de analizar los circuitos combinacionales con trenes de impulsos en sus entradas.

1. La salida de una puerta AND es un nivel ALTO sólo cuando todas las entradas están a nivel ALTO en el mismo instante.
2. La salida de una puerta OR es un nivel ALTO siempre que al menos una de sus entrada esté a nivel ALTO.
3. La salida de una puerta NAND es un nivel BAJO sólo cuando todas las entradas están a nivel ALTO en el mismo instante.
4. La salida de una puerta NOR es un nivel BAJO siempre que al menos una de las entradas esté a nivel ALTO.

EJEMPLO 5.10

Determinar la forma de onda de salida X para el circuito de la Figura 5.29, cuando se aplican las entradas A, B y C que se indican.

Solución La expresión de salida, $\overline{AB + AC}$, indica que la salida X es un nivel BAJO cuando A y B están a nivel ALTO, o cuando A y C están a nivel ALTO, o cuando todas las entradas están a nivel ALTO. En la Figura 5.29 se muestra el diagrama de tiempos correspondiente a la señal de salida X. La señal de salida intermedia Y, salida de la puerta OR, también se indica en el diagrama.

Problema relacionado Determinar la forma de onda de salida si la entrada A es un nivel ALTO constante.

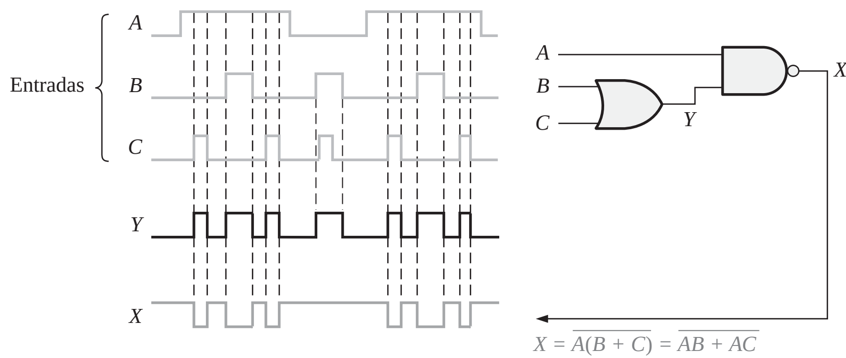


FIGURA 5.29

EJEMPLO 5.11

Dibujar el diagrama de tiempos para el circuito de la Figura 5.30, especificando las salidas de las puertas G_1, G_2 y G_3, siendo las entradas las formas de onda A y B que se indican.

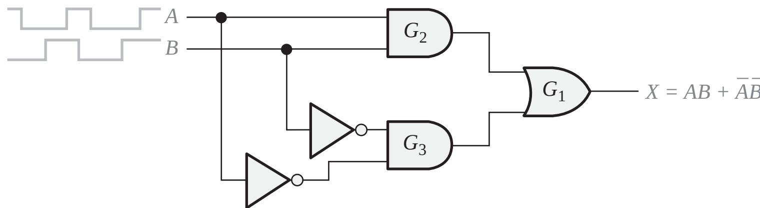


FIGURA 5.30

Solución Cuando las dos entradas están a nivel ALTO o a nivel BAJO, la salida X se pone a nivel ALTO, como muestra la Figura 5.31. Observe que se trata de un circuito NOR-exclusiva. Las salidas intermedias de las puertas G_2 y G_3 también se muestran en dicha figura.

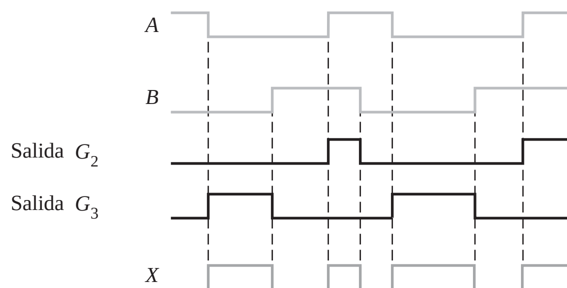


FIGURA 5.31

Problema relacionado Determinar la salida X correspondiente al circuito de la Figura 5.30, si se invierte la entrada B.

EJEMPLO 5.12

Determinar la forma de onda de salida X para el circuito lógico de la Figura 5.32(a), hallando en primer lugar las formas de onda intermedias en los puntos Y_1, Y_2, Y_3 e Y_4. Las formas de onda de entrada son las que se indican en la Figura 5.32(b).

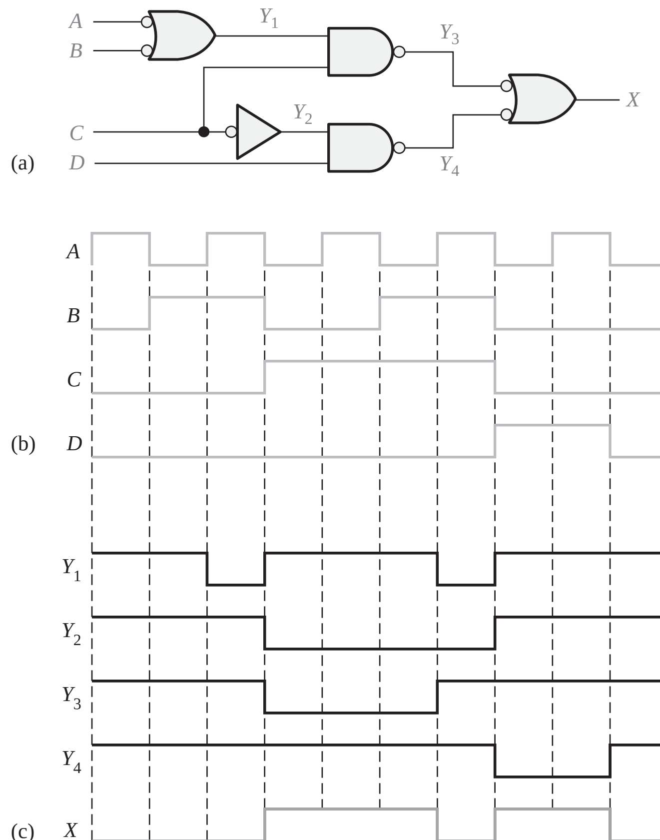


FIGURA 5.32

Solución Todas las formas de onda intermedias y la forma de onda de salida se muestran en el diagrama de tiempos de la Figura 5.32(c).

Problema relacionado Determinar las formas de onda Y_1, Y_2, Y_3 Y_4 y X si se invierte la entrada A.

EJEMPLO 5.13

Determinar la forma de onda de salida X para el circuito del Ejemplo 5.12, Figura 5.32(a), directamente a partir de la expresión de salida.

Solución En la Figura 5.33 se desarrolla la expresión de salida para el circuito especificado. La suma de productos indica que la salida es un nivel ALTO cuando A está a nivel BAJO y C está a nivel ALTO, o cuando B está a nivel BAJO y C está a nivel ALTO, o cuando C está a nivel BAJO y D está a nivel ALTO.

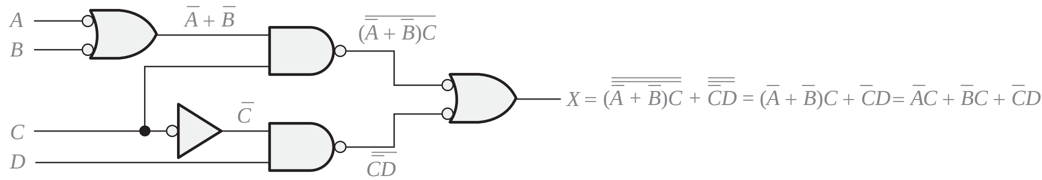


FIGURA 5.33

El resultado se muestra en la Figura 5.34, y es el mismo que se ha obtenido por el método de las señales intermedias del Ejemplo 5.12. Se indican los términos producto para cada forma de onda que da lugar a un nivel de salida ALTO.

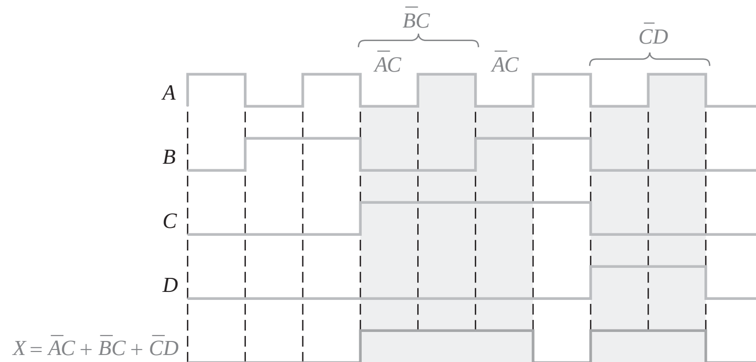


FIGURA 5.34

Problema relacionado Repetir este ejemplo si se invierten todas las formas de onda de entrada.

REVISIÓN DE LA SECCIÓN 5.5

1. A una de las entradas de un circuito OR-exclusiva se aplica un impulso con t_W = 50 μs. A la otra entrada, 15 μs después de que se genere el flanco de subida del primer impulso, se aplica un impulso positivo con t_W = 10 μs. Dibujar la salida en función de las entradas.
2. Los trenes de impulsos *A* y *B* de la Figura 5.29 se aplican al circuito NOR-exclusiva de la Figura 5.30. Desarrollar el diagrama de tiempos completo.

5.6 LÓGICA COMBINACIONAL CON VHDL (opcional)

El propósito de describir la lógica utilizando VHDL es que puede programarse en un PLD. En el Capítulo 4 se ha descrito el método del flujo de datos para escribir un programa VHDL. En esta sección opcional, vamos a usar tanto el método de flujo de datos usando expresiones booleanas como el método estructural para desarrollar el código VHDL que describe los circuitos lógicos. Vamos a presentar y a utilizar el componente VHDL para ilustrar las descripciones estructurales. También se abordan algunos aspectos de las herramientas de desarrollo software.

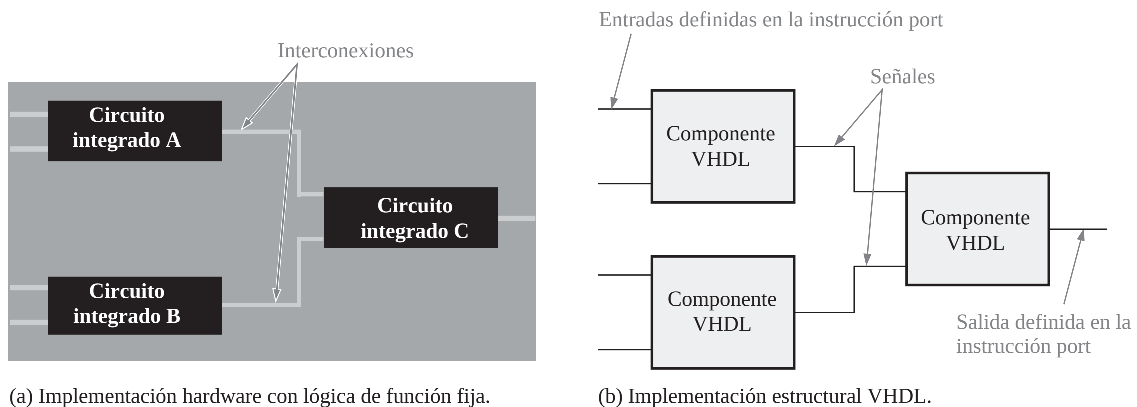
Al finalizar esta sección, el lector deberá ser capaz de:

■ Describir un componente VHDL y explicar cómo se emplea en un programa. ■ Aplicar el método estructural y el método de flujo de datos para escribir código VHDL. ■ Describir dos herramientas básicas de desarrollo software.

Método estructural de programación en VHDL

El método estructural para escribir una descripción VHDL de una función lógica puede compararse con el montaje de circuitos integrados en una tarjeta de circuito y el establecimiento de las interconexiones entre ellos mediante cables. Con el método estructural, se describen las funciones lógicas y se especifica cómo se conectan entre sí. El ***componente*** VHDL es una forma de predefinir una función lógica para poder emplearla repetidas veces en un mismo programa o en otros programas. El componente puede utilizarse para describir cualquier cosa, desde una simple puerta lógica a una función lógica compleja. La ***señal*** VHDL es una forma de especificar una conexión mediante un "cable" entre componentes.

La Figura 5.35 proporciona una comparación simplificada del método estructural con una implementación hardware en una tarjeta de circuito.



(a) Implementación hardware con lógica de función fija.

(b) Implementación estructural VHDL.

FIGURA 5.35 Comparación simplificada del método estructural VHDL con una implementación hardware. Las señales VHDL se corresponden con las interconexiones de la tarjeta de circuito y los componentes VHDL con los circuitos integrados.

Componentes VHDL

Un componente VHDL describe la lógica predefinida que puede almacenarse como una declaración de paquete en una biblioteca VHDL y puede llamarse tantas veces como sea necesario dentro de un programa. Puede emplear componentes para evitar repetir el mismo código una y otra vez dentro de un programa. Por ejemplo, puede crear un componente VHDL para una puerta AND y utilizarlo tantas veces como desee sin tener que escribir un programa para una puerta AND cada vez que lo necesite.

Los componentes VHDL se almacenan y están disponibles para su uso cuando se escribe un programa. Esto es similar, por ejemplo, a disponer de una bandeja de almacenamiento de circuitos integrados mientras se está montando un circuito. Cada que vez que se necesita un dispositivo integrado, se toma de la bandeja de almacenamiento y se coloca sobre la tarjeta de circuito impreso.

El programa VHDL para cualquier función lógica puede ser un componente y puede emplearse cuando sea necesario en un programa más largo mediante la declaración del componente, que tiene el formato general siguiente. **component** es una palabra clave VHDL.

component nombre_del_componente **is**
port (definiciones de puerto);
end component nombre_del_componente;

Para simplificar, como se muestra en la Figura 5.36 suponemos que tenemos descripciones de flujo de datos VHDL predefinidas para una puerta AND de 2 entradas con el nombre de entidad AND_gate y para una puerta OR de 2 entradas con el nombre de entidad OR_gate.

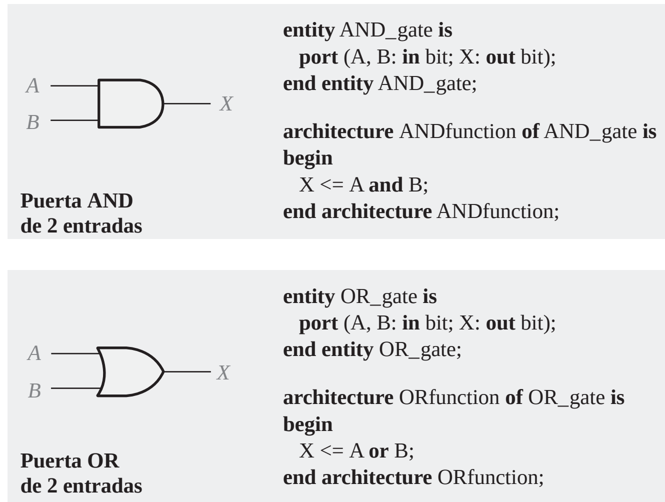


FIGURA 5.36 Programas predefinidos para una puerta AND de 2 entradas y para una puerta OR de 2 entradas que se emplearán como componentes en el método de flujo de datos.

A continuación suponemos que estamos escribiendo un programa para un circuito lógico que tiene varias puertas AND. En lugar de reescribir el programa de la Figura 5.36 una y otra vez podemos usar una declaración de componente para especificar la puerta AND. La instrucción *port* de la declaración del componente debe corresponderse con la instrucción *port* de la declaración de entidad de la puerta AND.

component AND_gate **is**
port (A, B: **in** bit; X: **out** bit);
end component AND_gate;

Utilización de componentes en un programa. Para emplear un componente en un programa, hay que escribir una instrucción de instantación de componente por cada instancia en la que se utilice el componente. Una instantación de componente es como una solicitud o llamada al componente que se va a usar en el programa principal. Por ejemplo, el circuito lógico en forma suma de productos simple de la Figura 5.37 tiene dos puertas AND y una puerta OR. Por tanto, el programa VHDL para este circuito tendrá dos componentes y tres instantaciones o llamadas a componente.

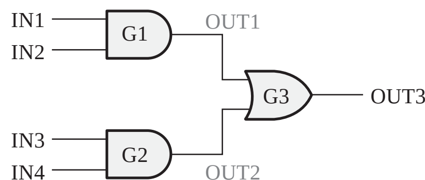


FIGURA 5.37

Señales. En VHDL, las señales son como los hilos que interconectan los componentes montados en una tarjeta de circuito impreso. En la Figura 5.37 las señales se han etiquetado como OUT1 y OUT2. Las señales son las conexiones *internas* del circuito lógico y se tratan de forma diferente que las entradas y salidas. Mientras que las entradas y salidas se declaran en la declaración de entidad utilizando la instrucción *port*, las señales se declaran dentro de la arquitectura mediante la instrucción *signal*, (**signal** es una palabra clave VHDL).

El programa. El programa para el circuito de la Figura 5.37 comienza con la siguiente declaración de entidad:

```
-- Programa para el circuito lógico de la Figura 5.37
entity AND_OR_Logic is
  port (IN1, IN2, IN3, IN4: in bit; OUT3: out bit);
end entity AND_OR_Logic;
```

La declaración de arquitectura contiene las declaraciones de los componentes para la puerta AND y la puerta OR, las definiciones de las señales y las instantaciones de los componentes.

```
architecture LogicOperation of AND_OR_Logic is

    component AND_gate is
      port (A, B: in bit); X: out bit);
    end component AND_gate;

    component OR_gate is
      port (A, B: in bit; X: out bit);
    end component OR_gate;

    signal OUT1, OUT2: bit;
begin
    G1: AND_gate port map (A => IN1, B => IN2, X => OUT1);
    G2: AND_gate port map (A => IN3, B => IN4, X => OUT2);
    G3: OR_gate port map (A => OUT1, B => OUT2, X => OUT3);
end architecture LogicOperation;
```

Declaración de componente para la puerta AND

Declaración de componente para la puerta OR

Declaración de las señales

Instantaciones de los componentes

Instantaciones de los componentes. Centrémonos en las instantaciones de los componentes. En primer lugar, fíjese en que las instantaciones de los componentes aparecen entre la palabra clave **begin** y la instrucción **end**. Para cada instantación, se define un identificador, en este caso, G1, G2 y G3. A continuación se especifica el nombre del componente. La instrucción *port map* establece, fundamentalmente, todas las conexiones de la función lógica utilizando el operador =>. Por ejemplo, la primera instantación,

```
G1: AND_gate port map (A => IN1, B => IN2, X => OUT1);
```

puede explicarse de la forma siguiente: *la entrada A de la puerta AND G1 se conecta a la entrada IN1, la entrada B de la puerta se conecta a la entrada IN2 y la salida X de la puerta se conecta a la señal OUT1.*

Las tres instrucciones de instantación describen completamente el circuito lógico de la Figura 5.37, como se ilustra en la Figura 5.38.

Aunque el método de flujo de datos usando expresiones booleanas hubiera sido más fácil y, probablemente, la mejor forma de describir este circuito concreto, hemos empleado este simple circuito para explicar el concepto del método estructural. El Ejemplo 5.14 compara los métodos de flujo de datos y estructural para escribir un programa VHDL para un circuito lógico en forma suma de productos.

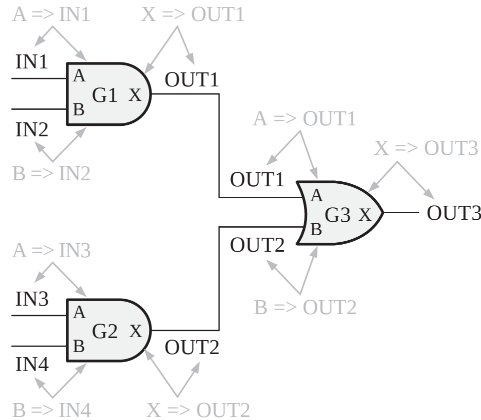


FIGURA 5.38 Ilustración de las instrucciones de instantación y la correspondencia de puertos (port map) aplicada a la lógica AND-OR. Las señales se indican en negro.

EJEMPLO 5.14

Escribir un programa VHDL para el circuito lógico en forma de suma de productos de la Figura 5.39 utilizando el método estructural. Suponemos que se dispone de los componentes para una puerta NAND de 3 entradas y para una puerta NAND de 2 entradas. Observe que la puerta NAND G4 se muestra como una puerta negativa-OR.

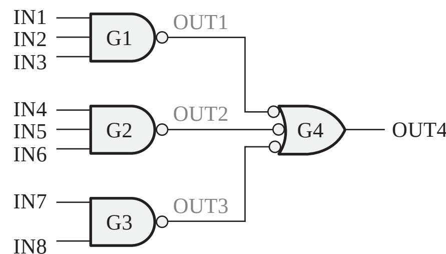


FIGURA 5.39

Solución Los componentes y las instantaciones de los componentes se muestran en negrita.

```
-- Programa para el circuito lógico de la Figura 5.39
entity SOP_Logic is
  port (IN1, IN2, IN3, IN4, IN5, IN6, IN7, IN8: in bit; OUT4: out bit);
end entity SOP_Logic;
architecture LogicOperation of SOP_Logic is
-- declaración de componente para una puerta NAND de 3 entradas
  component NAND_gate3 is
   port (A, B, C: in bit X: out bit);
  end component NAND_gate3;
```

```
-- declaración de componente para una puerta NAND de 2 entradas
  component NAND_gate2 is
    port (A, B: in bit; X: out bit);
  end component NAND_gate;
  signal OUT1, OUT2, OUT3: bit;
begin
  G1: NAND_gate3 port map (A => IN1, B => IN2, C => IN3, X => OUT1);
  G2: NAND_gate3 port map (A => IN4, B => IN5, C => IN6, X => OUT2);
  G3: NAND_gate2 port map (A => IN7, B => IN8, X => OUT3);
  G4: NAND_gate3 port map (A => OUT1, B => OUT2, C => OUT3,
            X => OUT4);
end architecture LogicOperation;
```

Con fines de comparación, escribimos el programa para el circuito lógico de la Figura 5.39 utilizando el método de flujo de datos.

```
entity SOP_Logic is
  port (IN1, IN2, IN3, IN4, IN5, IN6, IN7, IN8: in bit; OUT4: out bit);
end entity SOP_Logic;
architecture LogicOperation of SOP_Logic is
begin
  OUT4 <= (IN1 and IN2 and IN3) or (IN4 and IN5 and IN6) or (IN7 and IN8);
end architecture LogicOperation;
```

Como puede ver, el método de flujo de datos da lugar a un código mucho más sencillo en el caso de esta función lógica en concreto. Sin embargo, en situaciones en las que una función lógica consta de muchos bloques de lógica compleja, el método estructural puede resultar más ventajoso que el método de flujo de datos.

Problema relacionado Si se añade otra puerta NAND con entradas IN9 e IN10 al circuito de la Figura 5.39, escribir una instantación de componente para añadirla al programa. Especifique cualquier otro cambio que sea necesario realizar en el programa.

Aplicación de herramientas de desarrollo software

Como ya sabemos, debe utilizarse un paquete de desarrollo software para implementar un diseño HDL en un dispositivo objetivo. Una vez que la lógica se ha descrito empleando un lenguaje de descripción hardware y se ha introducido mediante una herramienta software, denominada editor de código o de texto, puede probarse mediante simulación con el fin de verificar que funciona correctamente antes de programar realmente el dispositivo objetivo. El uso de herramientas de desarrollo software permite diseñar, desarrollar y probar la lógica combinacional antes de implementarla en el hardware. En el Capítulo 11 se estudian más en detalle las herramientas de desarrollo software.

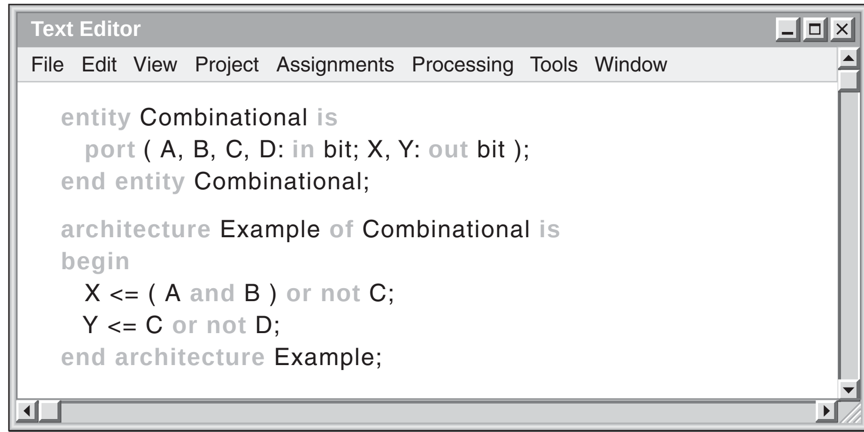


FIGURA 5.40 Programa VHDL para un circuito lógico combinacional como aparece en una pantalla de un editor de texto genérico, que forma parte de una herramienta de desarrollo software.

Las herramientas de desarrollo software típicas permiten introducir el código VHDL en un editor de texto específico de la herramienta de desarrollo concreta que se esté utilizando. En la Figura 5.40 se muestra una pantalla de computadora que contiene el código VHDL de un circuito lógico combinacional escrito en un editor de texto genérico. Como puede verse, muchos editores de código proporcionan funcionalidades mejoradas tales como resaltar las palabras clave.

Una vez que se ha escrito el programa en el editor de texto, se pasa al compilador. El compilador toma el código VHDL de alto nivel y lo convierte en un archivo que puede descargarse en el dispositivo objetivo. Una vez que el programa se ha compilado, se puede crear una simulación para probarlo. Los valores de entrada simulados se insertan en el diseño lógico y se puede proceder a la verificación de las salidas.

Las formas de onda de entrada se especifican mediante una herramienta software denominada editor de formas de onda, como la mostrada en la Figura 5.41. Las formas de onda de salida se generan mediante una simulación del código VHDL que se ha escrito en el editor de texto de la Figura 5.40. La simulación de la forma de onda proporciona las salidas resultantes *X* e *Y* para todas las combinaciones desde 0000_2 hasta 1111_2 de las entradas *A*, *B*, *C* y *D*.

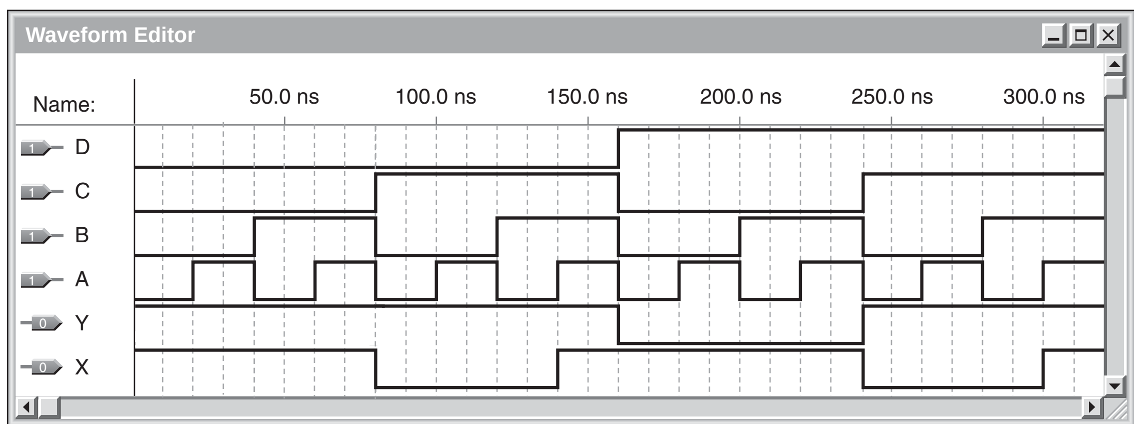


FIGURA 5.41 Un editor de formas de onda típico que muestra las formas de onda simuladas para el circuito lógico descrito por el código VHDL de la Figura 5.40.

Como se ha establecido en el Capítulo 3, no debe olvidar que en la creación de cualquier sistema digital deben tenerse en cuenta diversas características de funcionamiento de los circuitos lógicos. Por ejemplo, el retardo de propagación determina la velocidad o frecuencia a la que el circuito lógico puede funcionar. Puede utilizarse una simulación de temporización para simular el retardo de propagación a través del diseño lógico en el dispositivo objetivo.

REVISIÓN DE LA SECCIÓN 5.6

1. ¿Qué es un componente VHDL?
2. Explicar cuál es el propósito de una instantación de componente en una arquitectura de programa.
3. ¿Cómo se establecen las interconexiones entre los componentes en VHDL?
4. ¿Qué método representa el uso de componentes en un programa VHDL?

5.7 LOCALIZACIÓN DE AVERÍAS

Las secciones anteriores nos han introducido en el modo de operación de los circuitos lógicos combinacionales y las relaciones entre las entradas y las salidas. Estos conocimientos son esenciales cuando hay que localizar una avería en un circuito digital, ya que se debe conocer qué niveles lógicos o señales hay que buscar en el circuito para un conjunto dado de condiciones de entrada.

En esta sección, se utiliza un osciloscopio para localizar averías en un circuito lógico de función fija cuando una salida de una puerta está conectada a varias entradas de otras puertas. También se presenta un ejemplo de métodos de análisis y seguimiento de señales utilizando un oscilosopio o un analizador lógico para localizar un fallo en un circuito lógico combinacional.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Definir nodo de circuito. ■ Utilizar un osciloscopio para encontrar un fallo en un nodo de circuito. ■ Utilizar un osciloscopio para encontrar un circuito abierto en la salida de una puerta. ■ Utilizar un osciloscopio para encontrar un cortocircuito en la salida o la entrada de una puerta. ■ Utilizar un osciloscopio o un analizador lógico para seguir una señal en un circuito lógico combinacional.

En un circuito lógico combinacional, la salida de una puerta puede conectarse a dos o más entradas de otra puerta, como muestra la Figura 5.42. Las interconexiones se cruzan en un punto eléctrico común que se denomina **nodo**.

La puerta G_1 de la Figura 5.42 excita al nodo, y las demás puertas representan las cargas conectadas al nodo. Una puerta excitadora puede excitar a un determinado número de entradas de puertas de carga, hasta el máximo determinado por su *fan-out* específico. En esta situación, se pueden producir diversos tipos de fallos. Algunos de estos tipos de fallos son difíciles de aislar en una puerta, ya que todas las puertas conectadas al nodo se ven afectadas. Los fallos más comunes son los siguientes:

1. *Salida en circuito abierto en la puerta excitadora*. Este fallo da lugar a pérdida de la señal en todas las puertas de carga.
2. *Entrada en circuito abierto en una puerta de carga*. Este fallo no afectará al funcionamiento de ninguna otra puerta conectada al nodo, pero hará que no se detecte señal de salida en la puerta que falla.
3. *Salida cortocircuitada de la puerta excitadora*. Este fallo puede dar lugar a que el nodo permanezca en estado BAJO (cortocircuitado a masa) o en estado ALTO (cortocircuitado a V_{CC}).
4. *Entrada cortocircuitada en una puerta de carga*. Este fallo también hace que el nodo se mantenga a nivel BAJO (cortocircuitado a masa) o en estado ALTO (cortocircuitado a V_{CC}).

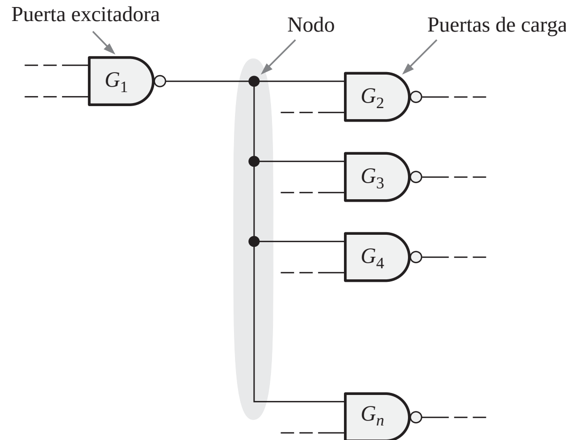


FIGURA 5.42 Ilustración de un nodo en un circuito lógico.

CONSEJOS PRÁCTICOS

Cuando se están localizando averías en circuitos lógicos, se debe empezar por una comprobación visual con el fin de localizar los problemas obvios. La inspección visual debería incluir los conectores además de los componentes. Los conectores externos se usan frecuentemente para llevar a una tarjeta de circuito las señales, la alimentación y la masa. Las superficies de contacto de los conectores deben estar limpias y deben tener una buena fijación mecánica. Un conector sucio puede producir un fallo intermitente o completo del circuito. Los conectores externos se pueden limpiar con un borrador de lápiz normal y un bastoncillo humedecido en alcohol. También se deberían comprobar todos los conectores para localizar los pines que no estén bien ajustados.

Localización de los fallos más comunes

Salida en circuito abierto en la puerta excitadora. En este caso no se detectan impulsos en el nodo. Con el circuito alimentado, un nodo en circuito abierto dará lugar, normalmente, a un nivel "flotante", lo que puede indicarse mediante ruido, como se ilustra en la Figura 5.43.

Entrada en circuito abierto en una puerta de carga. Si la salida de la puerta excitadora no está en circuito abierto, entonces hay que probar si la entrada de una puerta de carga está en circuito abierto. Permaneciendo las entradas de las puertas no pertenecientes al nodo a nivel ALTO, se comprueba con el osciloscopio la salida de cada una de las puertas, como se indica en la Figura 5.44. Si una de las entradas conectadas al nodo está en circuito abierto, no se detectarán impulsos en la salida de la misma.

Salida o entrada cortocircuita a masa. Si la salida está cortocircuitada a masa en la puerta excitadora o la entrada a una puerta de carga está cortocircuitada a masa, esto hará que el nodo permanezca a nivel BAJO, como se ha dicho anteriormente. Una rápida comprobación con la sonda del osciloscopio lo detectará, como se muestra en la Figura 5.45. Un cortocircuito a masa en la salida de la puerta excitadora o en la entrada de cualquier puerta de carga dará lugar a este síntoma, por lo que deben hacerse más comprobaciones para aislar el cortocircuito en una puerta en concreto.

Análisis y seguimiento de señales

Aunque los métodos de aislamiento de cortocircuitos y circuitos abiertos en un nodo son muy útiles en ocasiones, una técnica más general para la localización de fallos es el ***seguimiento de señales***, la cual tiene un

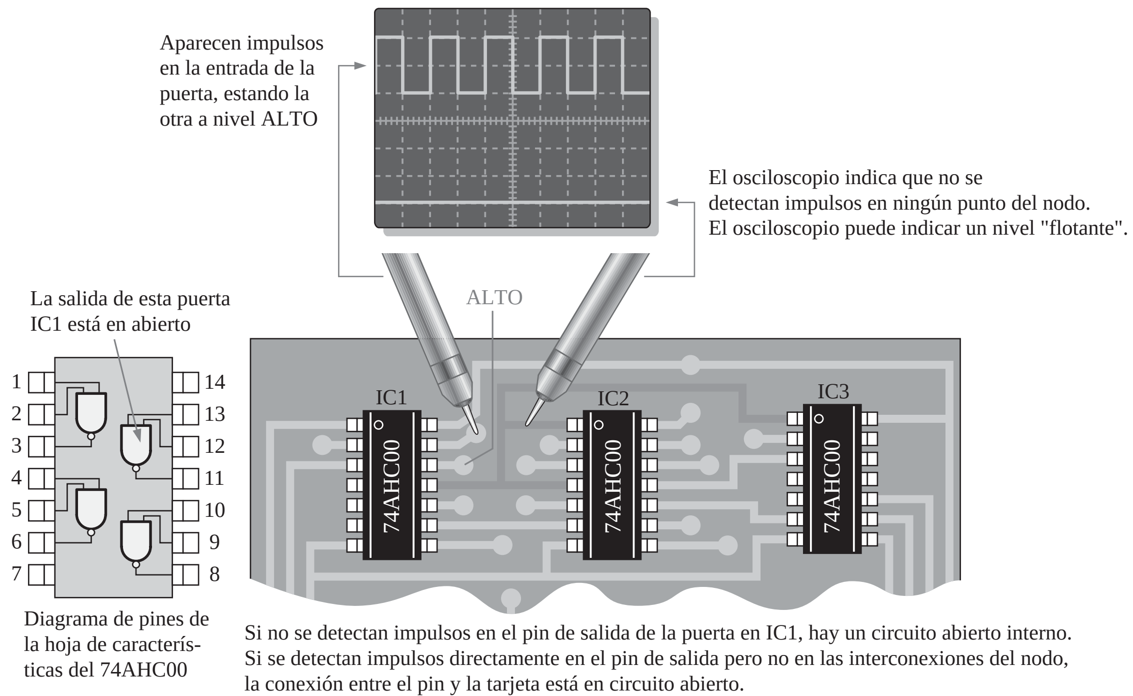


FIGURA 5.43 Salida en circuito abierto en la puerta excitadora. Para simplificar, se supone que hay un nivel ALTO en la entrada de una puerta.

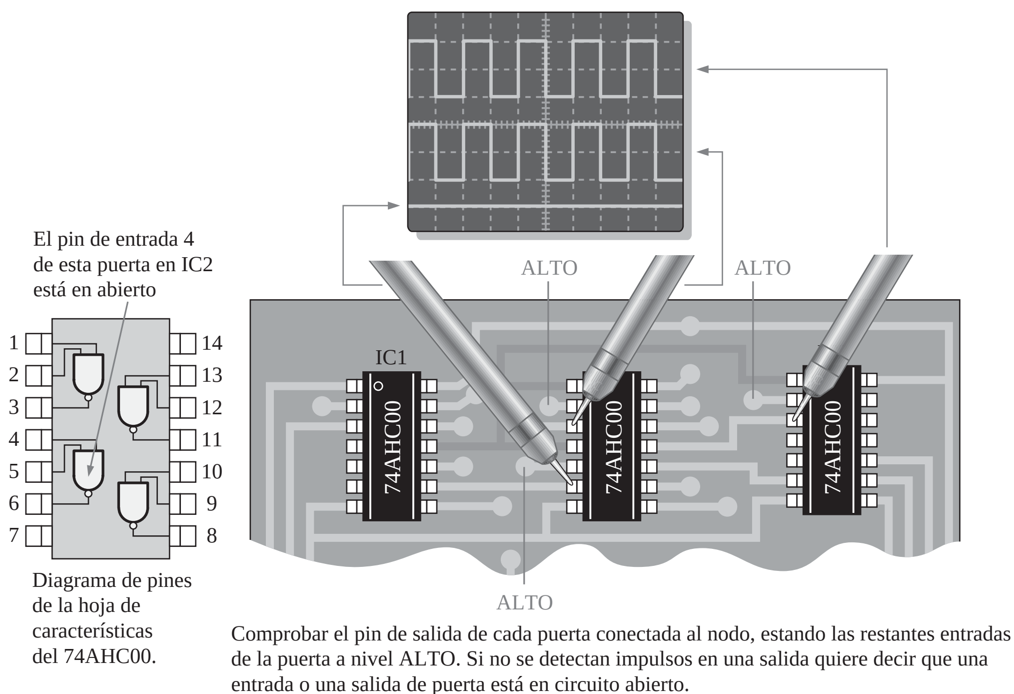


FIGURA 5.44 Entrada en circuito abierto de una puerta de carga.

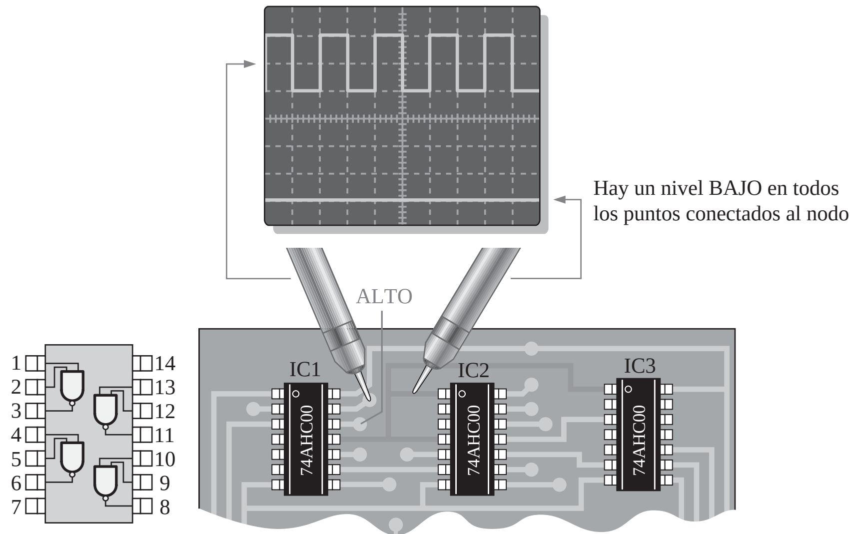


FIGURA 5.45 Salida cortocircuitada en una puerta excitadora o entrada cortocircuitada en una puerta de carga.

gran valor para el técnico en casi todos los casos de fallo. La medida de la señal se realiza con un osciloscopio o un analizador lógico.

Básicamente, el método de seguimiento de señales requiere que se observen las formas de onda y sus relaciones temporales en todos los puntos accesibles del circuito lógico. Se puede comenzar por las entradas y, a partir del análisis del diagrama de tiempos de la señal en cada punto, determinar cuál es el primer punto en que la señal es incorrecta. Normalmente, con este procedimiento se puede aislar el fallo en una puerta específica. También se puede usar el método de comenzar por la salida y continuar hacia atrás hasta las entradas.

El procedimiento general del seguimiento de señales comenzando por las entradas es el siguiente:

- Dentro del sistema, definir la sección del circuito lógico que se sospecha que está fallando.
- Comenzar en las entradas de la sección que se va a examinar. Para este estudio, suponemos que las formas de onda de entrada proceden de otras partes del sistema que son correctas.
- Para cada puerta, empezando por la entrada y yendo hacia la salida del circuito lógico, se observa la forma de onda de salida de la puerta y se compara con las formas de onda de entrada, utilizando el osciloscopio o el analizador lógico.
- Determinar si la señal de salida es correcta utilizando nuestros conocimientos sobre la operación lógica de la puerta.
- Si la salida es incorrecta, en la puerta bajo prueba puede estar el fallo. Extraiga el CI que contiene la puerta de la que se sospecha que produce el fallo, y compruébelo fuera del circuito. Si la puerta falla, reemplace el CI. Si funciona correctamente, el fallo está en la circuitería externa o en otro CI al que está conectado el que se está probando.
- Si la salida es correcta, pase a la puerta siguiente. Continúe comprobando cada puerta hasta observar una forma de onda incorrecta.

La Figura 5.46 es un ejemplo que ilustra el procedimiento general para un circuito lógico específico, en el que se siguen los pasos que a continuación se indican:

Paso 1. Observar la salida de la puerta G_1 (punto de prueba 5) respecto a sus entradas. Si es correcta, probar el inversor siguiente. Si la salida no es correcta, la puerta o sus conexiones están mal; o, si la salida está a nivel BAJO, la entrada de la puerta G_2 puede estar cortocircuitada.

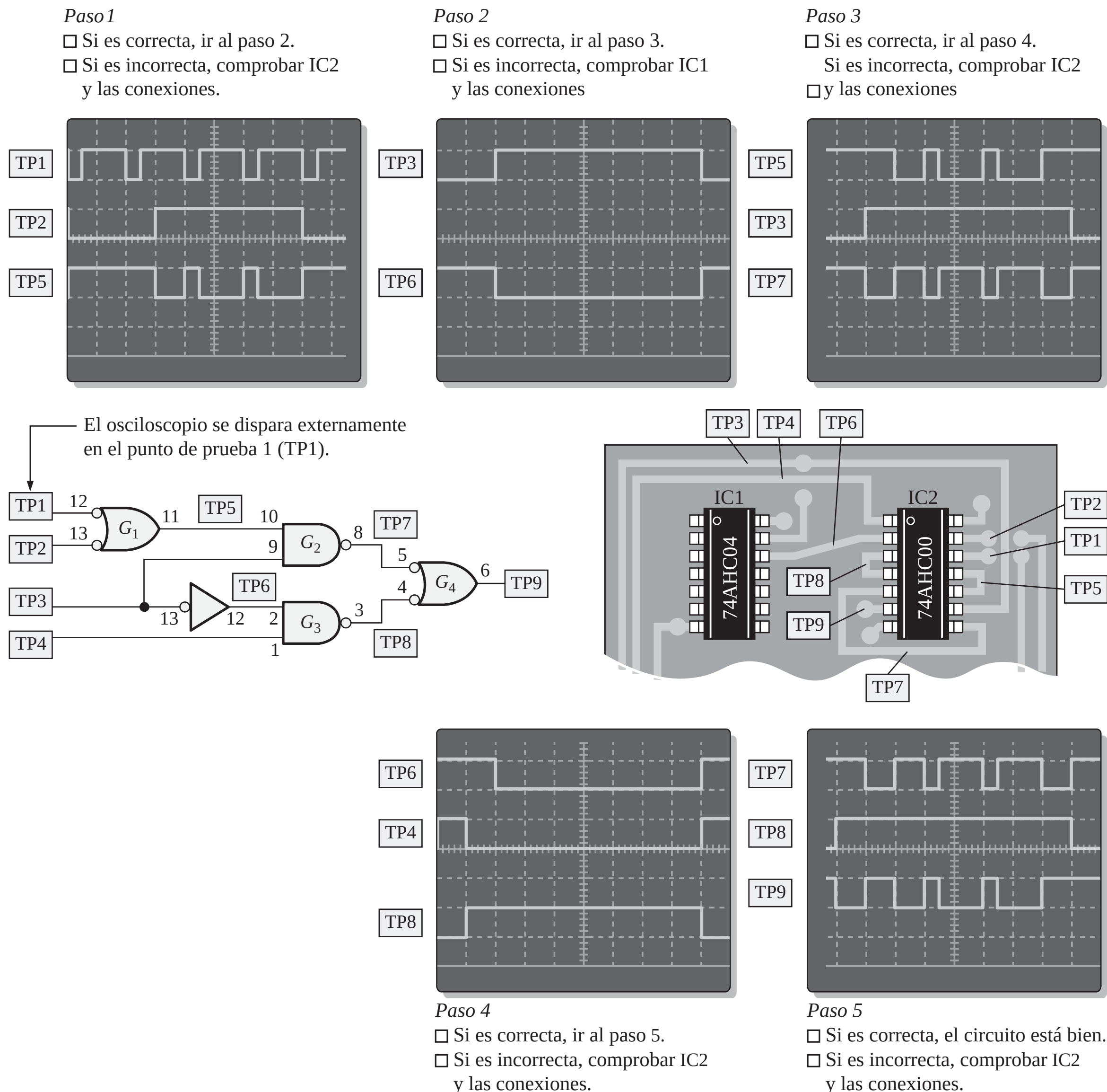


FIGURA 5.46 Ejemplo de análisis y seguimiento de señales en una parte de una tarjeta de circuito impreso. TP (test point) indica punto de prueba.

Paso 2. Observar la salida del inversor (TP6) respecto a la entrada. Si es correcta, probar la puerta siguiente, G_2. Si la salida no es correcta, el inversor o sus conexiones están mal; o, si la salida está a nivel BAJO, la entrada de la puerta G_3 puede estar cortocircuitada.

Paso 3. Observar la salida de la puerta G_2 (TP7) respecto a las entradas. Si es correcta, probar la puerta siguiente, G_3. Si la salida no es correcta, el inversor o sus conexiones están mal; o, si la salida está a nivel BAJO, la entrada de la puerta G_4 puede estar cortocircuitada.

Paso 4. Observar la salida de la puerta G_3 (TP8) respecto a sus entradas. Si es correcta, probar la puerta G_4. Si la salida no es correcta, la puerta o sus conexiones están mal; o, si la salida está a nivel BAJO, la entrada de la puerta G_4 (TP9) puede estar cortocircuitada.

Paso 5. Observar la salida de la puerta G_4 (TP9) respecto a sus entradas. Si es correcta, el circuito está bien. Si la salida no es correcta, la puerta o sus conexiones están mal.

EJEMPLO 5.15

Determinar el fallo en el circuito lógico de la Figura 5.47(a) utilizando el análisis de señales. Debe observar las formas de onda de color gris de la Figura 5.47(b). Las formas de onda en negro son correctas y se proporcionan con propósitos de comparación.

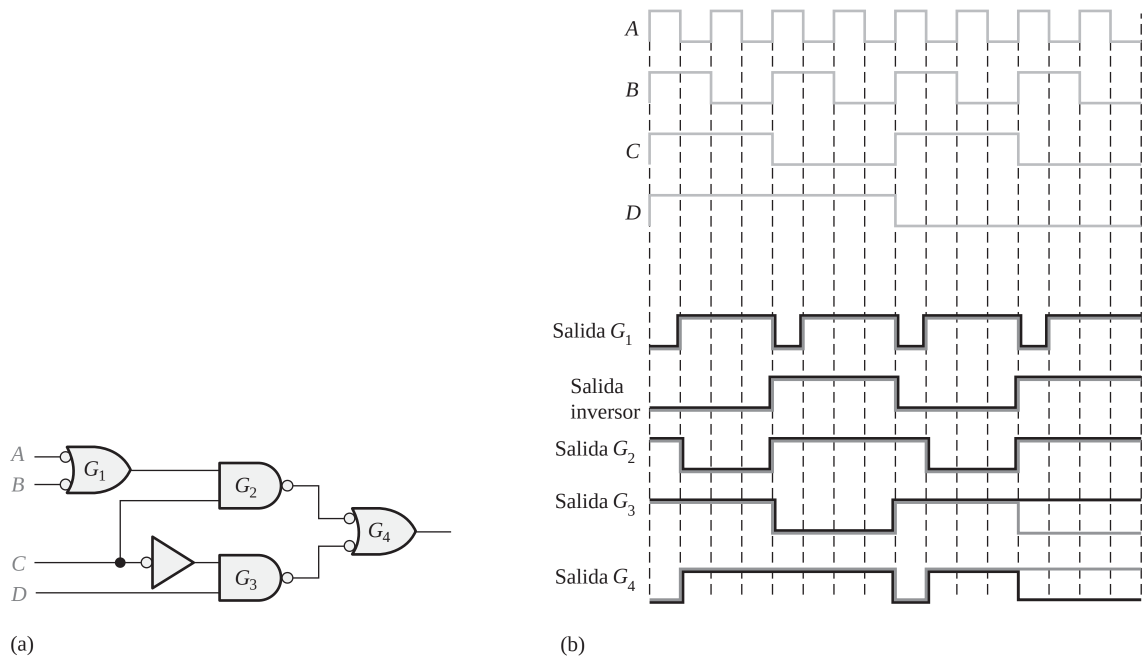


FIGURA 5.47

Solución

1. Determinar cuál es la forma de onda correcta para cada puerta. Las formas de onda correctas se muestran en color negro en la Figura 5.47(b), sobreimpresas a las formas de onda medidas realmente.
2. Comparar las formas de onda puerta por puerta hasta encontrar una señal medida que no se corresponda con la señal correcta.

En este ejemplo, todas las comprobaciones son correctas hasta llegar a la puerta G_3. La salida de ésta es incorrecta de acuerdo con las diferencias que se indican. Un análisis de la señal pone de manifiesto que, si la entrada D de la puerta G_3 está en circuito abierto y opera como un nivel ALTO, se obtendría la forma de onda de salida medida (mostrada en negro). Observe que la salida de G_4 también es incorrecta ya que la entrada procedente de G_3 es incorrecta.

Reemplace el circuito integrado que contiene a G_3, y pruebe de nuevo el funcionamiento del circuito.

Problema relacionado Para las entradas de la Figura 5.47(b), determinar la señal de salida del circuito lógico (salida de G_4), si la salida del inversor está en circuito abierto.

CONSEJOS PRÁCTICOS

Como ya sabemos, probar y localizar los fallos en circuitos lógicos frecuentemente requiere observar y comparar dos formas de onda digitales de forma simultánea, tal como una entrada y la salida de una puerta, en un osciloscopio de doble canal. Para las formas de onda digitales, el osciloscopio siempre debería configurarse con acoplamiento DC en cada entrada de canal, para evitar los "desplazamientos" del nivel de tierra. Deberá determinar dónde se encuentra el nivel de 0 V en la pantalla para ambos canales.

Para comparar la temporización de las formas de onda, el osciloscopio debería dispararse sólo desde un canal (no utilice el disparo en modo vertical o compuesto). El canal seleccionado para disparo debería ser siempre aquél que tenga la frecuencia más baja, cuando sea posible.

REVISIÓN DE LA SECCIÓN 5.7

1. Enumerar cuatro tipos de fallos comunes en las puertas lógicas.
2. Una entrada de una puerta NOR está externamente cortocircuitada a $+V_{CC}$. ¿Cómo afecta esta condición al funcionamiento de la puerta?
3. Determinar la salida de la puerta G_4 de la Figura 5.47(a), si se aplican las entradas de la Figura 5.47(b), para los fallos siguientes:
   (a) Una entrada de G_1 cortocircuitada a masa.
   (b) La entrada del inversor cortocircuitada a masa.
   (c) Una salida en circuito abierto en G_3.



APLICACIÓN A LOS SISTEMAS DIGITALES

En esta aplicación a los sistemas digitales, se desarrolla la lógica de control de un sistema digital que permite controlar el fluido que hay en un tanque de almacenamiento. El propósito de la lógica es mantener un nivel apropiado de fluido controlando las válvulas de entrada y de salida. La lógica también tiene que controlar la temperatura del fluido dentro de un determinado rango y disparar una alarma si el sensor de nivel o el sensor de temperatura falla.

Funcionamiento básico del sistema

Las salidas de la lógica de control del sistema controlan la entrada de fluido, la salida de fluido y la temperatura del mismo. La lógica de control actúa sobre una válvula de entrada que permite que el fluido entre en el tanque hasta que el sensor de nivel alto se activa al quedar sumergido en el fluido. Cuando el sensor de nivel alto está sumergido (activado) la lógica de control cierra la válvula de entrada. El fluido contenido en el tanque debe mantenerse dentro de un rango de temperatura especificado, el cual queda determinado por dos sensores de temperatura. Uno de los sensores de temperatura indica si el fluido está demasiado caliente y el otro si el fluido está demasiado frío. La lógica de control activa un elemento de calefacción si los sensores de temperatura indican que el fluido está demasiado

frío. La lógica de control mantiene abierta la válvula de salida siempre que el sensor de nivel bajo esté sumergido y el fluido se encuentre a la temperatura adecuada. Cuando el nivel de fluido cae por debajo del sensor de nivel bajo, la lógica de control cierra la válvula de salida.

Requisitos de operación

Los niveles máximo y mínimo de fluido quedan determinados por las posiciones de los sensores de nivel del tanque. La salida de cada sensor estará a nivel ALTO mientras que esté sumergido en el fluido y estará a nivel BAJO cuando no quede sumergido. Cuando la salida del sensor de nivel alto está a nivel BAJO, la lógica de control genera un nivel ALTO y abre la válvula de entrada. Cuando la salida del sensor de nivel alto está a nivel ALTO, la lógica de control genera un nivel BAJO y cierra la válvula de entrada.

Antes de abrir la válvula de salida, el fluido debe encontrarse dentro del rango de temperatura especificado. Un sensor genera un nivel ALTO cuando el fluido está muy caliente y el otro sensor de temperatura genera un nivel ALTO cuando la temperatura es demasiado baja. La lógica de control genera un nivel ALTO para activar el elemento de calefacción cuando se tiene la indicación de temperatura baja; en caso contrario, el elemento de calefacción está apagado. Cuando aparece la condición de temperatura alta, se activa una alarma.

Cuando el sensor de nivel bajo genera una salida a nivel ALTO (lo que indica que está sumergido) y la salida de los dos sensores de temperatura están a nivel BAJO (lo que indica que el fluido está a la temperatura correcta), la lógica de control abre la válvula de salida. Si la salida del sensor de nivel bajo pasa a nivel BAJO o si las salidas de los sensores de temperatura pasan a nivel ALTO, la lógica de control cierra la válvula de salida.

Si la lógica de control detecta un fallo en cualquiera de los sensores o una condición de temperatura muy alta, se activa una alarma. Un fallo en un sensor de nivel se produce cuando el sensor de nivel alto está activado y el sensor de nivel bajo no lo está. Un fallo en un sensor de temperatura se indica mediante la activación de los dos sensores a un mismo tiempo. La Figura 5.48 muestra el sistema de control del tanque.

En la Tabla 5.6 se resumen las entradas y las salidas del sistema y en la Tabla 5.7 se ilustra la tabla de verdad.

Diseño de la lógica de control

Hay cuatro salidas diferentes, una para la válvula de entrada, una para la válvula de salida, una para el sistema de calefacción y una para la alarma. Vamos a abordar el diseño como cuatro circuitos lógicos separados.

Lógica de la válvula de entrada Comenzamos diseñando el circuito lógico para la válvula de entrada. La salida de este circuito lógico es la variable V_{ENTRADA}. El primer paso consiste en transferir los datos de la tabla de verdad a un mapa de Karnaugh y desarrollar una expresión suma de productos.

Las variables de entrada L_H, L_L, T_H y T_L son las variables del mapa y los estados de V_{ENTRADA} se dibujan y agrupan como se muestra en la Figura 5.49(a). Los 0s del mapa son las condiciones de entrada cuando la válvula de entrada está cerrada y los 1s son las condiciones de entrada

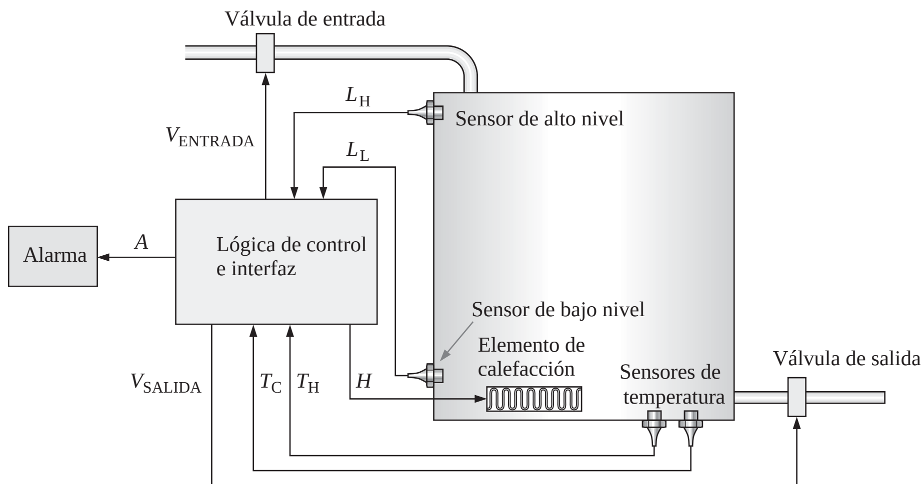


FIGURA 5.48 Tanque de almacenamiento de fluido con controles y sensores de nivel y de temperatura.

ENTRADAS DE LA LÓGICA DE CONTROL			
Variable	**Descripción**	**Nivel activo**	**Comentarios**
L_H	Sensor de nivel alto	ALTO (1)	El sensor está sumergido
L_L	Sensor de nivel bajo	ALTO (1)	El sensor está sumergido
T_H	Sensor de temperatura alta	ALTO (1)	Temperatura muy alta
T_L	Sensor de temperatura baja	ALTO (1)	Temperatura muy baja
SALIDAS DE LA LÓGICA DE CONTROL			
Variable	**Descripción**	**Nivel activo**	**Comentarios**
$V_{ENTRADA}$	Válvula de entrada	ALTO (1)	Válvula abierta
V_{SALIDA}	Válvula de salida	ALTO (1)	Válvula abierta
H	Elemento de calefacción	ALTO (1)	Calefacción activada
A	Alarma	ALTO (1)	Fallo de sensor o condición de temperatura alta

TABLA 5.6 Entradas y salidas de la lógica de control del tanque.

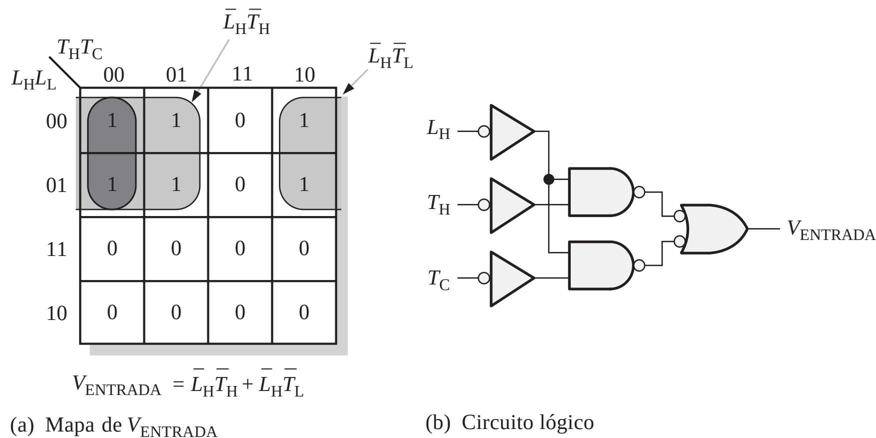
Entradas				Salidas				
L_H	L_L	T_H	T_L	$V_{ENTRADA}$	V_{SALIDA}	H	A	**Comentarios**
0	0	0	0	1	0	0	0	Rellenar/calefacción apagada
0	0	0	1	1	0	1	0	Rellenar/calefacción encendida
0	0	1	0	1	0	0	1	Rellenar/calefacción apagada/alarma
0	0	1	1	0	0	0	1	Fallo sensor de temp./alarma
0	1	0	0	1	1	0	0	Rellenar y vaciar/ calefacción apagada
0	1	0	1	1	0	1	0	Rellenar/calefacción encendido
0	1	1	0	1	0	0	1	Rellenar/calefacción apagada/alarma
0	1	1	1	0	0	0	1	Fallo sensor de temp./alarma
1	0	0	0	0	0	0	1	Fallo sensor de nivel/alarma
1	0	0	1	0	0	0	1	Fallo sensor de nivel/alarma
1	0	1	0	0	0	0	1	Fallo sensor de nivel/alarma
1	0	1	1	0	0	0	1	Fallo de varios sensores /alarma
1	1	0	0	0	1	0	0	Vaciar/calefacción apagada
1	1	0	1	0	0	1	0	Calefacción encendida
1	1	1	0	0	0	0	1	Calefacción apagada/alarma
1	1	1	1	0	0	0	1	Fallo de sensor de temp/alarma

TABLA 5.7 Tabla de verdad para la lógica de control del tanque.

cuando dicha válvula está abierta. La expresión suma de productos resultante para la lógica de la válvula de entrada da lugar a la implementación NAND mostrada en la parte (b) de la figura.

Lógica de la válvula de salida A continuación diseñamos el circuito lógico para la válvula de salida. La salida de este circuito lógico es V_{SALIDA}. De nuevo, el primer paso consiste en transferir los datos de la tabla de verdad a un mapa de Karnaugh y desarrollar una expresión suma de productos.

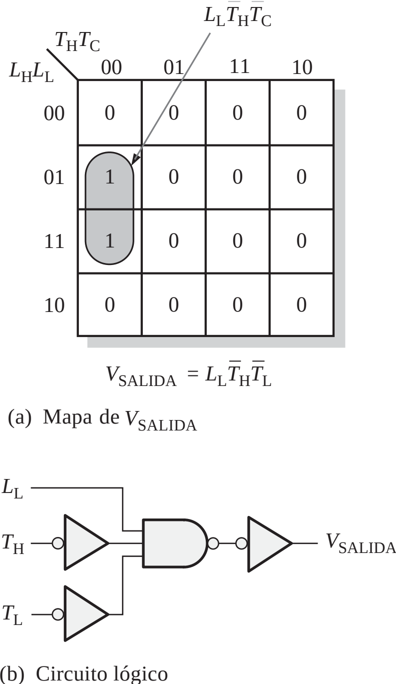
La variables de entrada, L_H, L_L, T_H y T_L son las variables del mapa y los estados de V_{SALIDA} se dibujan y agrupan



(a) Mapa de $V_{ENTRADA}$ (b) Circuito lógico

FIGURA 5.49 Simplificación mediante el mapa de Karnaugh e implementación de la lógica de la válvula de entrada.

como se muestra en la Figura 5.50(a). Los 0s del mapa son las condiciones de entrada cuando la válvula de salida está cerrada y los 1s son las condiciones de entrada cuando dicha válvula está abierta. La expresión suma de productos resultante para la lógica de la válvula de salida da lugar a la implementación NAND mostrada en la parte (b).



(a) Mapa de V_{SALIDA}

(b) Circuito lógico

FIGURA 5.50 Simplificación mediante el mapa de Karnaugh e implementación de la lógica de la válvula de salida.

Código VHDL para la lógica de las válvulas de entrada y de salida (opcional)

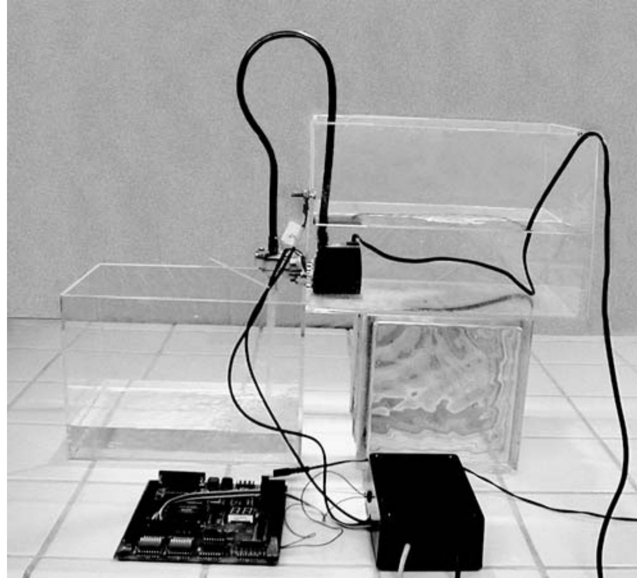
Una misma entidad y arquitectura describen la lógica de la válvula de entrada y la lógica de la válvula de salida utilizando el método de flujo de datos, como muestra el siguiente programa.

```
entity TankControl is
  port (LL, LH, TH, TL in bit; Ventrada,
    Vsalida: out bit);
end entity TankControl;
architecture ValveLogic of TankControl is
begin
    Ventrada <= (not LH and not TH) or
      (not LH and not TL);
    Vsalida <= LL and not TH and not TL;
end architecture ValveLogic;
```

Se ha diseñado y desarrollado el código VHDL para la lógica de las válvulas de entrada y de salida. Ahora, es el momento de que complete el resto del diseño de la lógica de control para el elemento de calefacción y la alarma, y de que escriba el programa VHDL para implementar la lógica en un dispositivo objetivo.

Práctica de sistemas

- ***Actividad 1.*** Utilizando la Tabla 5.7 y el método de mapa de Karnaugh, diseñar la lógica para controlar el



Una fotografía de una réplica de un tanque de almacenamiento del laboratorio de electrónica del Yuba College en California. La lógica de control se ha programado en un PLD sobre una tarjeta de desarrollo y se ha conectado al tanque para controlar el llenado y el vaciado del mismo. Cortesía de Doug Joksch.

elemento de calefacción del tanque. Utilice puertas NAND e inversores para implementar el circuito.

- ***Actividad 2.*** Diseñar la lógica para activar la alarma.
- ***Actividad 3.*** Combinar la lógica de cada una de las cuatro funciones de control del tanque en un diagrama lógico completo.
- ***Actividad opcional.*** Escribir la arquitectura y la entidad VHDL para la lógica completa, modificando el código anteriormente desarrollado para la lógica de las válvulas de entrada y de salida.

RESUMEN

- La lógica AND-OR genera una expresión de salida en forma de suma de productos.
- La lógica AND-OR_Inversor genera una forma suma de productos complementada, la cual realmente es una forma producto de sumas.
- El símbolo operacional para la operación OR-Exclusiva es $\oplus$. Una expresión OR-Exclusiva puede expresarse de dos formas equivalentes

$$A\overline{B} + \overline{A}B = A \oplus B$$

- Para hacer un análisis de un circuito lógico, se parte del circuito lógico y se desarrolla la expresión de salida booleana o la tabla de verdad, o ambas.
- La implementación de un circuito lógico es el proceso por el que, partiendo de las expresiones booleanas de salida o de la tabla de verdad, se desarrolla un circuito que genera la función de salida.
- Todos los diagrama lógicos NAND y NOR deben dibujarse empleando los símbolos duales apropiados, de modo que las salidas invertidas (con círculo) se conecten a entradas invertidas y las salida no invertidas (sin círculo) se conecten a entradas no invertidas.
- Cuando se conectan dos indicadores de negación (círculos), se cancelan entre sí.

- Un componente VHDL es una función lógica predefinida que se almacena para utilizarla a lo largo de un programa o en otros programas.
- Una instantación de componente se utiliza para llamar a un componente en un programa.
- Una señal VHDL actúa como una interconexión interna en una descripción estructural VHDL.

PALABRAS CLAVE

Las palabras clave y otros términos que se han resaltado en negrita se encuentran en el glosario final del libro.

Componente Una funcionalidad VHDL que puede utilizarse para predefinir una función lógica que puede emplearse a lo largo de un programa o programas.
Negativa-AND La operación dual de una puerta NOR cuando las entradas son activas a nivel BAJO.
Negativa-OR La operación dual de una puerta NAND cuando las entradas son activas a nivel BAJO.
Nodo Punto de conexión común en un circuito, en el que la salida de una puerta se conecta a una o más entradas de puerta.
Puerta universal Tanto una puerta NAND como NOR. El término *universal* se refiere a la propiedad de aquellas puertas que permiten que cualquier operación lógica pueda ser implementada mediante ellas o mediante una combinación de puertas de ese tipo.
Seguimiento de señales Técnica de localización de averías mediante la cual se observan las señales paso a paso, comenzando en la entrada y siguiéndolas hasta la salida, o viceversa. En cada punto, las formas de ondas observadas se comparan con la señal correcta que debería haber en ese punto.
Señal Una forma de onda; un tipo de objeto VHDL que almacena datos.

AUTOTEST

Las respuestas se encuentran al final del capítulo.

1. La expresión de salida para un circuito AND-OR que tiene una puerta AND con las entradas A, B, C y D y una puerta AND con las entradas E y F es

(a) $ABCDEF$ **(b)** $A + B + C + D + E + F$

(c) $(A + B + C + D)(E + F)$ **(d)** $ABCD + EF$

2. Un circuito lógico con una salida $A\overline{B}C + A\overline{C}$ consta de

(a) Dos puertas AND y una puerta OR

(b) Dos puertas AND, una puerta OR y dos inversores

(c) Dos puertas OR, una puerta AND y dos inversores

(d) Dos puertas AND, una puerta OR y un inversor.

3. Para implementar la expresión $\overline{A}BCD + A\overline{B}CD + ABC\overline{D}$, se necesita una puerta OR y

(a) una puerta AND

(b) tres puertas AND

(c) tres puertas AND y cuatro inversores

(d) tres puertas AND y tres inversores

4. La expresión $\overline{A}BCD + ABC\overline{D} + A\overline{B}\overline{C}D$

(a) no puede simplificarse

(b) puede simplificarse a $\overline{A}BC + A\overline{B}$

(c) puede simplificarse a $ABC\overline{D} + \overline{A}B\overline{C}$

(d) ninguna de las respuestas anteriores es correcta

5. La expresión de salida de un circuito AND-OR_Inversor que tiene una puerta AND con entradas A, B, C y D y una puerta AND con entradas E y F es

(a) $ABCD + EF$

(b) $\overline{A} + \overline{B} + \overline{C} + \overline{D} + \overline{E} + \overline{F}$

(c) $\overline{(A + B + C + D)(E + F)}$

(d) $(\overline{A} + \overline{B} + \overline{C} + \overline{D})(\overline{E} + \overline{F})$

6. Una expresión OR-Exclusiva se expresa como

(a) $\overline{A}\overline{B} + AB$

(b) $\overline{A}B + A\overline{B}$

(c) $(\overline{A} + B)(A + \overline{B})$

(d) $(\overline{A} + \overline{B}) + (A + B)$

7. La operación AND se puede generar con

(a) dos puertas NAND

(b) tres puertas NAND

(c) una puerta NOR

(d) tres puertas NOR

8. La operación OR se puede generar con

(a) dos puertas NOR **(b)** tres puertas NAND

(c) cuatro puerta NAND **(d)** las respuestas (a) y (b)

9. Cuando se usan símbolos duales en un diagrama lógico

(a) las salida invertidas (con círculo) se conectan a las entradas invertidas (con círculo)

(b) los símbolos NAND generan las operaciones AND

(c) los símbolos negativa-OR generan las operaciones OR

(d) todas las respuestas son verdaderas

(e) ninguna respuesta es verdadera

10. Todas las expresiones booleanas pueden implementarse con

(a) sólo puertas NAND

(b) sólo puertas NOR

(c) combinaciones de puertas NAND y NOR

(d) combinaciones de puertas AND, puertas OR e inversores

(e) todas las anteriores

11. Un componente VHDL

(a) puede usarse sólo una vez en cada programa

(b) es una descripción predefinida de una función lógica

(c) puede utilizarse múltiples veces en un programa

(d) es parte de una descripción de flujo de datos

(e) las respuestas (b) y (c)

12. En un programa, se llama a un componente utilizando

(a) una señal **(b)** una variable

(c) una instantación de componente **(d)** una declaración de arquitectura

PROBLEMAS

Las respuestas a los problemas impares se encuentran al final del libro.

SECCIÓN 5.1 Circuitos lógicos combinacionales básicos

1. Dibujar el diagrama lógico con símbolos distintivos ANSI de un circuito AND-OR-Inversor de 4 entradas y triple anchura. Dibujar también el diagrama utilizando los símbolos rectangulares ANSI estándar.
2. Escribir la expresión de salida de los circuitos de la Figura 5.51.
3. Escribir la expresión de salida de los circuitos de la Figura 5.52.

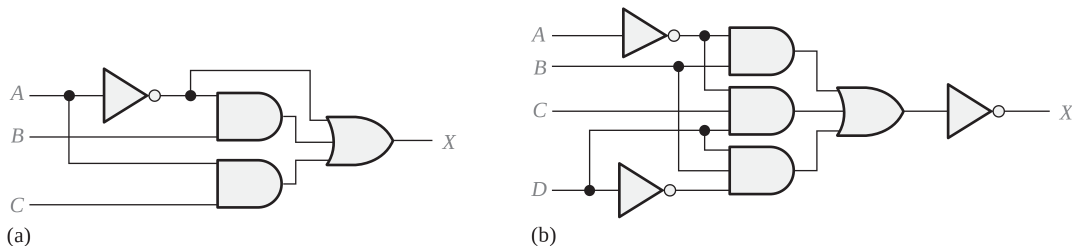


FIGURA 5.51

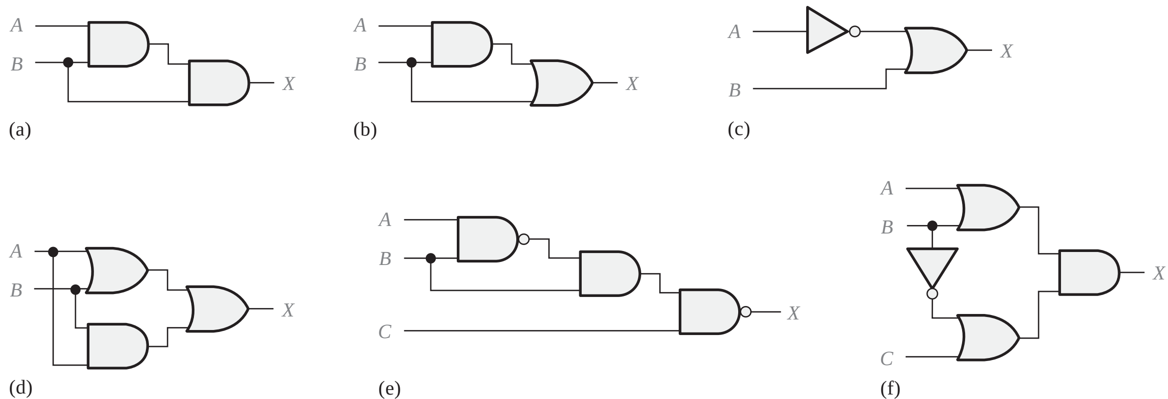


FIGURA 5.52

4. Escribir la expresión de salida de los circuitos de la Figura 5.53, y dibujar los circuitos equivalentes utilizando la configuración AND-OR.
5. Desarrollar la tabla de verdad de cada uno de los circuitos de la Figura 5.52.
6. Desarrollar la tabla de verdad de cada uno de los circuitos de la Figura 5.53.
7. Demostrar que un circuito NOR-exclusiva genera una salida que es un producto de sumas.

SECCIÓN 5.2 Implementación de la lógica combinacional

8. Utilizando puertas AND, puertas OR o combinaciones de ambas, implementar las siguientes expresiones lógicas:

   (a) $X = AB$

   (b) $X = A + B$

   (c) $X = AB + C$

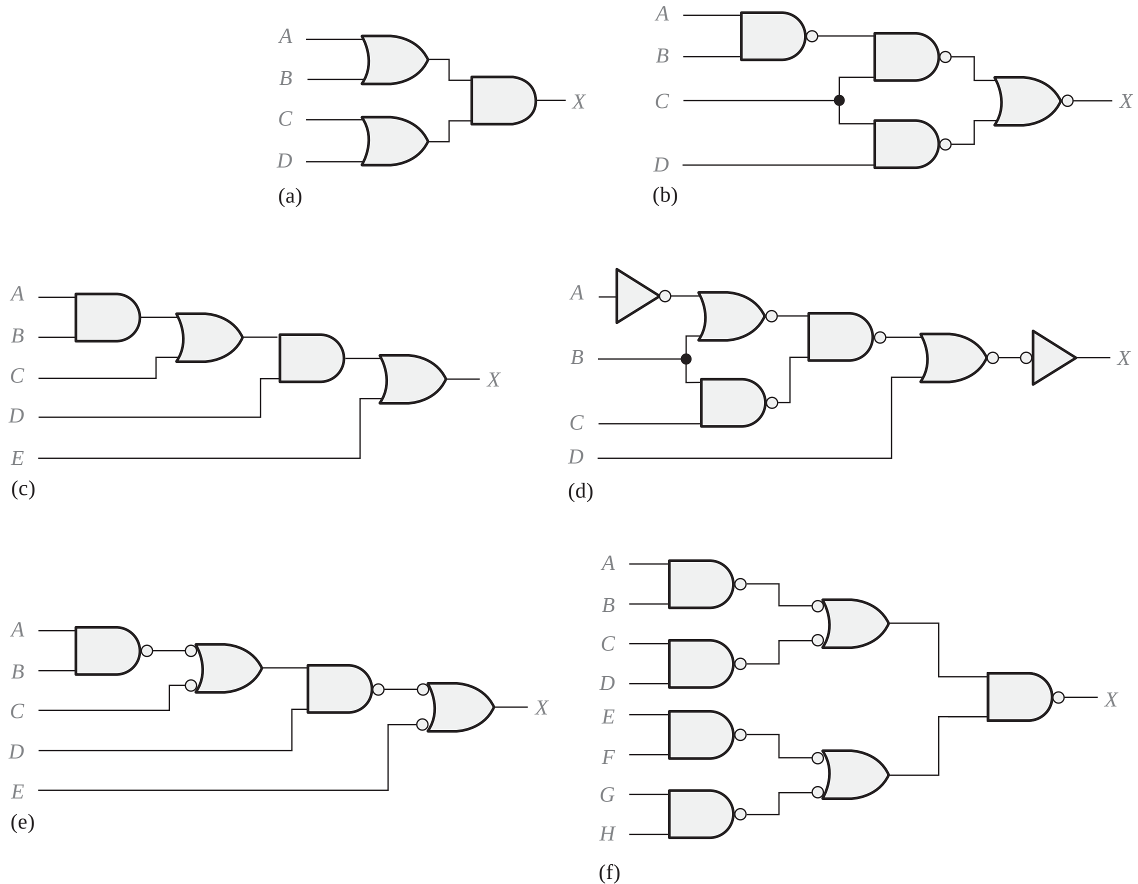


FIGURA 5.53

(d) $X = ABC + D$

(e) $X = A + B + C$

(f) $X = ABC\,D$

(g) $X = A(CD + B)$

(h) $X = AB(C + DEF) + CE(A + B + F)$

9. Utilizando puertas AND, puertas OR e inversores cuando sea necesario, implementar las siguientes expresiones lógicas:

(a) $X = AB + \overline{B}C$ **(b)** $X = A(B + \overline{C})$

(c) $X = A\overline{B} + AB$ **(d)** $X = \overline{ABC} + B(EF + \overline{G})$

(e) $X = A[BC(A + B + C + D)]$ **(f)** $X = B(C\overline{D}E + \overline{E}FG)(\overline{AB} + C)$

10. Utilizando puertas NAND, puertas NOR o combinaciones de ambas, implementar las siguientes expresiones lógicas:

(a) $X = \overline{A}B + CD + (\overline{A + B})(ACD + \overline{BE})$

(b) $X = AB\overline{C}\overline{D} + D\overline{E}F + \overline{AF}$

(c) $X = A[B + \overline{C}(D + E)]$

11. Implementar un circuito lógico para la tabla de verdad de la Tabla 5.8.

Entradas			Salida
A	*B*	*C*	*X*
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

TABLA 5.8

12. Implementar un circuito lógico para la tabla de verdad de la Tabla 5.9.

Entradas				Salida
A	*B*	*C*	*D*	*X*
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

TABLA 5.9

13. Simplificar el circuito de la Figura 5.54 tanto como sea posible, y verificar que el circuito simplificado es equivalente al original, demostrando que las tablas de verdad son idénticas.

14. Repetir el Problema 13 para el circuito de la Figura 5.55.

15. Minimizar las puertas requeridas para implementar las funciones de cada apartado del Problema 9 en forma de suma de productos.

16. Minimizar las puertas requeridas para implementar las funciones de cada apartado del Problema 10 en forma de suma de productos.

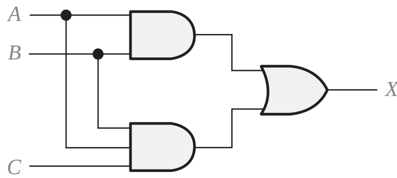


FIGURA 5.54

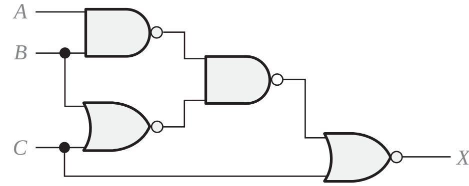


FIGURA 5.55

17. Minimizar las puertas requeridas para implementar la función de los circuitos de cada apartado de la Figura 5.53 en forma de suma de productos.

SECCIÓN 5.3 La propiedad universal de las puertas NAND y NOR

18. Implementar los circuitos lógicos de la Figura 5.51 utilizando sólo puertas NAND.

19. Implementar los circuitos lógicos de la Figura 5.55 utilizando sólo puertas NAND.

20. Repetir el Problema 18 utilizando sólo puertas NOR.

21. Repetir el Problema 19 utilizando sólo puertas NOR.

SECCIÓN 5.4 Lógica combinacional con puertas NAND y NOR

22. Mostrar cómo pueden implementarse las siguientes expresiones utilizando sólo puertas NOR:

(a) $X = ABC$ **(b)** $X = \overline{ABC}$ **(c)** $X = A + B$

(d) $X = A + B + \overline{C}$ **(e)** $X = \overline{AB} + \overline{CD}$ **(f)** $X = (A + B)(C + D)$

(g) $X = AB[C(\overline{DE} + \overline{AB}) + \overline{BCE}]$

23. Repetir el Problema 23 utilizando sólo puertas NAND.

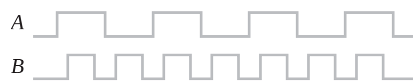
24. Implementar cada una de las funciones del Problema 8 utilizando sólo puertas NAND.

25. Implementar cada una de las funciones del Problema 9 utilizando sólo puertas NAND.

SECCIÓN 5.5 Funcionamiento de los circuitos lógicos con trenes de impulsos

26. Dados el circuito lógico y las formas de onda de entrada de la Figura 5.56, dibujar la forma de onda de salida.

27. Para el circuito lógico de la Figura 5.57, dibujar la forma de onda de salida con respecto a las entradas.



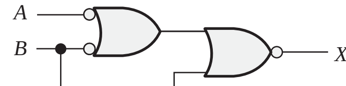
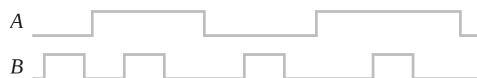


FIGURA 5.56



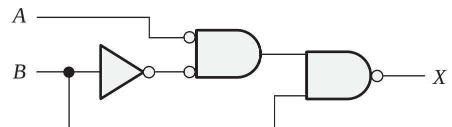


FIGURA 5.57

28. Para las formas de onda de entrada de la Figura 5.58, ¿qué circuito lógico generará la señal de salida mostrada?

29. Repetir el Problema 28 para la señal de la Figura 5.59.

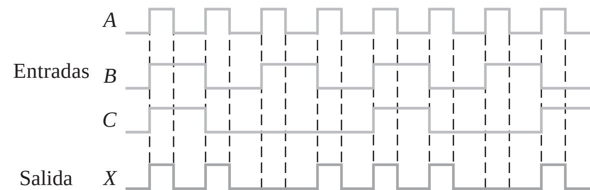


FIGURA 5.58

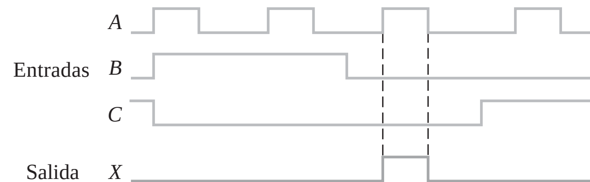


FIGURA 5.59

30. Para el circuito de la Figura 5.60, dibujar las formas de onda para los puntos numerados, indicando la relación de tiempos entre ellos.

31. Suponiendo un tiempo de propagación en cada puerta de 10 nanosegundos (ns), determinar si las entradas indicadas generarán la forma de onda de salida *X deseada* de la Figura 5.61 (impulso con un mínimo t_W = 25 ns como el mostrado).

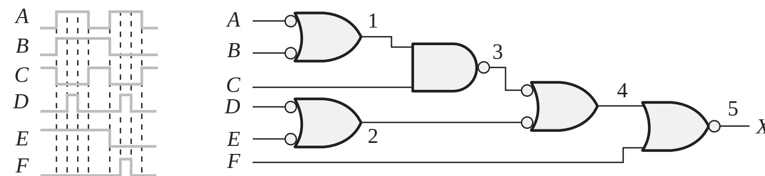
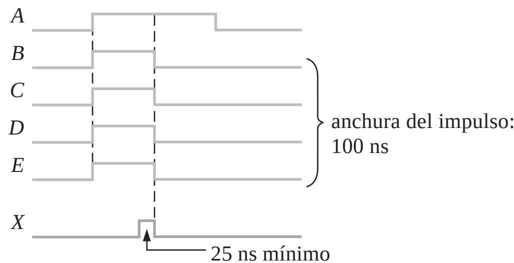


FIGURA 5.60



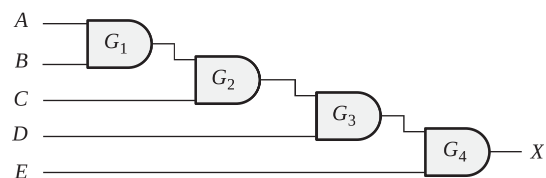


FIGURA 5.61

SECCIÓN 5.6 Lógica combinacional con VHDL (opcional)

32. Escribir un programa VHDL usando el método de flujo de datos (expresiones booleanas) para describir el circuito lógico de la Figura 5.51(b).

33. Escribir un programa VHDL usando el método de flujo de datos (expresiones booleanas) para describir los circuitos lógicos de las Figuras 5.52(e) y (f).

34. Escribir un programa VHDL usando el método estructural para describir el circuito lógico de la Figura 5.53(d). Suponer que están disponibles las declaraciones de componentes para cada tipo de puerta.

35. Repetir el Problema 34 para el circuito lógico de la Figura 5.53(f).

36. Describir la lógica representada por la tabla de verdad de la Tabla 5.8 utilizando VHDL, pasándola primero a una forma suma de productos.

37. Desarrollar un programa VHDL para la lógica de la Figura 5.64, utilizando los métodos de flujo de datos y estructural. Comparar los programas resultantes.

38. Desarrollar un programa VHDL para la lógica de la Figura 5.68, utilizando los métodos de flujo de datos y estructural. Comparar los programas resultantes.

39. Dado el siguiente programa VHDL, crear la tabla de verdad que describe el circuito lógico.

```
entity CombLogic is
  port (A, B, C, D: in bit; X: out bit);
end entity CombLogic;
architecture Example of CombLogic is
  begin
    X <= not((not A and not B) or (not A and not C) or (not A and not D) or
         (not B and not C) or(not B and not D) or (not D and not C));
end architecture Example;
```

40. Describir el circuito lógico mostrado en la Figura 5.62 con un programa VHDL, utilizando el método de flujo de datos.

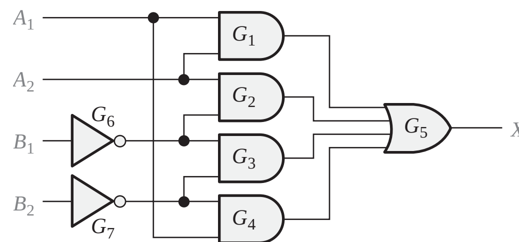


FIGURA 5.62

41. Repetir el Problema 40 utilizando el método estructural.

SECCIÓN 5.7 Localización de averías

42. Para el circuito lógico y la señal de entrada de la Figura 5.63, se observa la señal de salida indicada. Determinar si esta señal de salida es correcta.

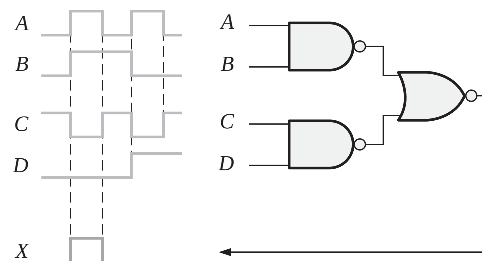


FIGURA 5.63

43. La forma de onda de salida de la Figura 5.64 es incorrecta para las entradas que se aplican al circuito. Suponiendo que una puerta del circuito está fallando, con su salida a un nivel ALTO o BAJO constante, determinar la puerta que falla y el tipo de fallo (circuito abierto o cortocircuito).

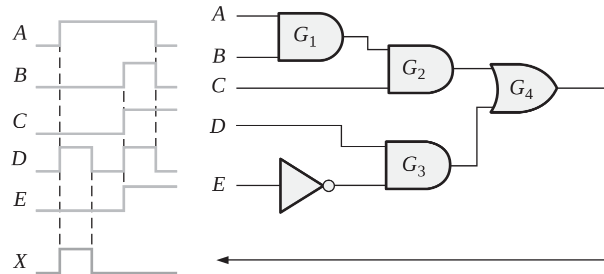


FIGURA 5.64

44. Repetir el Problema 43 para el circuito de la Figura 5.65 para las señales de entrada y salida dadas.

45. Examinando las conexiones de la Figura 5.66, determinar la puerta excitadora y las puertas de carga. Especificar por dispositivo y números de pines.

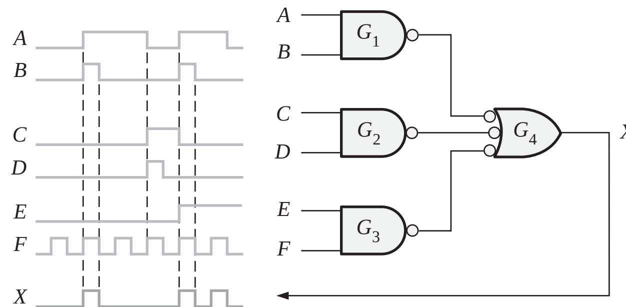


FIGURA 5.65

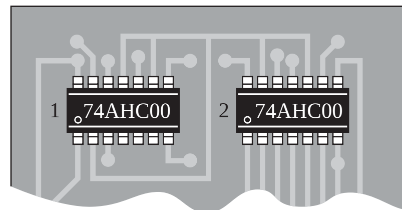


FIGURA 5.66

46. La Figura 5.67(a) es un circuito lógico bajo prueba. La Figura 5.67(b) muestra las formas de onda que se observan en el analizador lógico. Para las entradas que se aplican al circuito, la salida es incorrecta. Suponiendo que una puerta del circuito ha fallado, estando su salida a un nivel ALTO o a nivel BAJO constante, determinar la puerta que falla y el tipo de fallo.

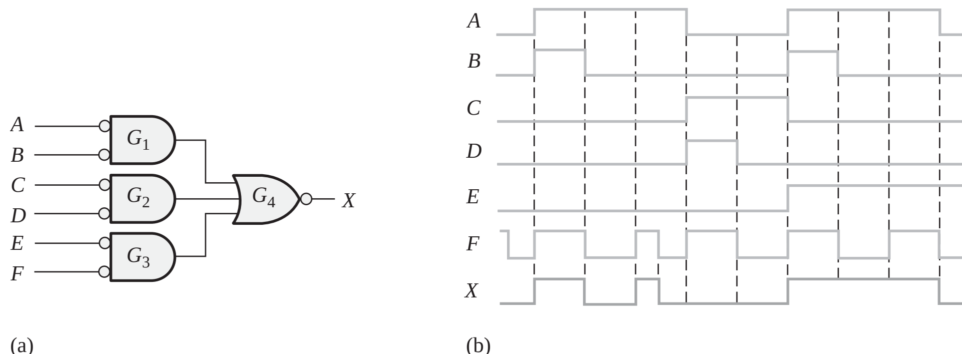


FIGURA 5.67

47. Al circuito lógico de la Figura 5.68 se le aplican las formas de onda de entrada mostradas.

(a) Determinar la señal de salida correcta con respecto a las entradas.

(b) Determinar la señal de salida si la salida de la puerta G_3 está en circuito abierto.

(c) Determinar la señal de salida si la entrada superior de la puerta G_3 está cortocircuitada a masa.

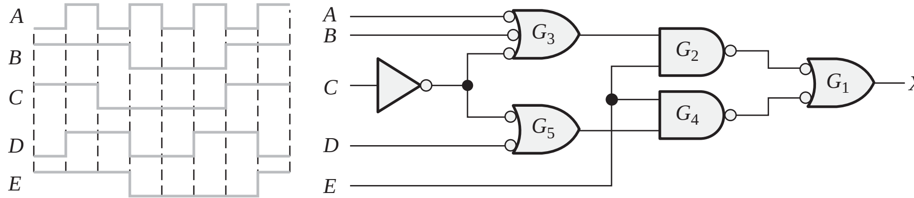


FIGURA 5.68

48. El circuito lógico de la Figura 5.69 tiene disponible un único punto de prueba intermedio próximo a la salida. Para las entradas indicadas, se observa la señal dada en el punto de prueba. ¿Es correcta esta forma de onda? Si no lo es, ¿cuáles son los posibles fallos que podrían generar dicha señal?

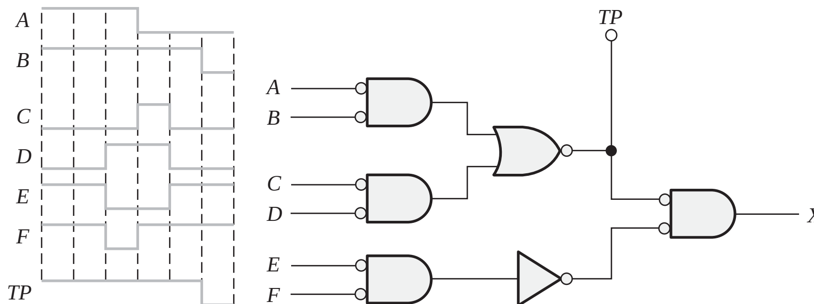


FIGURA 5.69

ICONO APLICACIONES A SISTEMAS



Aplicación a los sistemas digitales

49. Implementar la lógica de la válvula de entrada de la Figura 5.49(b) con puertas NOR e inversores.

50. Repetir el Problema 49 para la lógica de la válvula de salida de la Figura 5.50(b).

51. Implementar la lógica del elemento de calefacción y de la alarma usando puertas NOR e inversores.



Problemas especiales de diseño

52. Diseñar un circuito lógico para generar una salida a nivel ALTO si y sólo si la entrada, representada por un número binario de 4 bits, es mayor que doce o menor que tres. Desarrolle primero la tabla de verdad y después dibuje el diagrama lógico.

53. Desarrollar el circuito lógico que cumpla los siguientes requisitos:

Una lámpara situada en una habitación puede accionarse mediante dos interruptores, uno colocado detrás de la puerta y el otro frente a la puerta. La lámpara se enciende si se activa el interruptor frente a la puerta y el de detrás de la misma no se activa, o en el caso contrario. La lám-

para está apagada si ambos interruptores están desactivados o si ambos están activados. Una salida a nivel ALTO representa una condición de encendido y una salida a nivel BAJO representa la condición de apagado.

54. Diseñar un circuito que permite introducir un aditivo químico en el fluido a través de otra válvula de entrada sólo cuando la temperatura no sea ni demasiado baja ni demasiado alta y el fluido se encuentre por encima del sensor de nivel alto.

55. Desarrollar el diagrama lógico NAND para un codificador hexadecimal de teclado que convierta cada pulsación a binario.

RESPUESTAS

REVISIONES DE CADA SECCIÓN

SECCIÓN 5.1 Circuitos lógicos combinacionales básicos

1. **(a)** $\overline{AB+CD} = \overline{1 \cdot 0 + 1 \cdot 0} = 1$
   (b) $\overline{AB+CD} = \overline{1 \cdot 1 + 0 \cdot 1} = 0$
   (c) $\overline{AB+CD} = \overline{0 \cdot 1 + 1 \cdot 1} = 0$

2. **(a)** $A\overline{B}+\overline{A}B = 1 \cdot \overline{0} + \overline{1} \cdot 0 = 1$
   (b) $A\overline{B}+\overline{A}B = 1 \cdot \overline{1} + \overline{1} \cdot 1 = 0$
   (c) $A\overline{B}+\overline{A}B = 0 \cdot \overline{1} + \overline{0} \cdot 1 = 1$
   (d) $A\overline{B}+\overline{A}B = 0 \cdot \overline{0} + \overline{0} \cdot 0 = 0$

3. $X = 1$ cuando $ABC = 000$, 011, 101, 110 y 111. $X = 0$ cuando $ABC = 001$, 010 y 100.

4. $X = AB + \overline{A}\overline{B}$; el circuito está constituido por dos puertas AND, una puerta OR y dos inversores. Consulte el diagrama de la Figura 5.6(b).

SECCIÓN 5.2 Implementación de la lógica combinacional

1. **(a)** $X = ABC + AB + AC$; tres puertas AND, una puerta OR.
   (b) $X = AB(C + DE)$; tres puertas AND, una puerta OR.

2. $X = ABC + \overline{A}\overline{B}\overline{C}$; dos puertas AND, una puerta OR y tres inversores.

3. **(a)** $X = AB(C + 1) + AC = AB + AC$
   (b) $X = AB(C + DE) = ABC + ABDE$

SECCIÓN 5.3 La propiedad universal de las puertas NAND y NOR

1. **(a)** $X = \overline{A} + B$: es una puerta NAND con A y $\overline{B}$ en sus entradas.
   (b) $X = A\overline{B}$: es una puerta NAND con A y $\overline{B}$ en sus entradas, seguida de una puerta NAND utilizada como inversor.

2. **(a)** $X = \overline{A} + B$: es una puerta NOR con $\overline{A}$ y B en sus entradas, seguida de una puerta NOR utilizada como inversor.
   (b) $X = A\overline{B}$: es una puerta NOR con $\overline{A}$ y B en sus entradas.

SECCIÓN 5.4 Lógica combinacional con puertas NAND y NOR

1. $X = \overline{(\overline{A}+\overline{B}+\overline{C})DE}$: una puerta NAND de 3 entradas con las entradas A, B y C, con su salida conectada a una segunda puerta NAND de 3 entradas con otras dos entradas D y E.

2. $X = \overline{\overline{A}\overline{B}\overline{C} + (D+E)}$: una puerta NOR de 3 entradas con las entradas A, B y C, con su salida conectada a una segunda puerta NOR de 3 entradas con otras dos entradas D y E.

SECCIÓN 5.5 Funcionamiento de los circuitos lógicos con trenes de impulsos

1. La salida de la puerta OR-exclusiva es un impulso de 15 μs seguido de un impulso de 25 μs, con una separación de 10 μs entre los impulsos.
2. La salida de la puerta NOR-exclusiva es un nivel ALTO cuando ambas entradas están a nivel ALTO, o cuando ambas entradas están a nivel BAJO.

SECCIÓN 5.6 Lógica combinacional con VHDL (opcional)

1. Un componente VHDL es un programa predefinido que describe una función lógica especificada.
2. Una instantación de componente se utiliza para llamar a un componente especificado en una arquitectura de programa.
3. Las interconexiones entre componentes se hacen utilizando señales VHDL.
4. Los componentes se emplean en el método estructural.

SECCIÓN 5.7 Localización de averías

1. Los fallos más comunes en las puertas son entrada o salida en circuito abierto, y entrada o salida cortocircuitada a masa.
2. La entrada cortocircuitada a V_{CC} hace que la salida se mantenga a nivel BAJO.
3. **(a)** La salida de G_4 está a nivel alto hasta el flanco de bajada del séptimo impulso; luego pasa a nivel bajo.

   (b) La salida de G_4 es igual a la entrada D.

   (c) La salida de G_4 es la misma que la salida de G_2, mostrada en la Figura 5.47(b).

PROBLEMAS RELACIONADOS

5.1 $X = AB + AC + BC$

5.2 $X = \overline{AB + AC + BC}$

$$\text{Si } A = 0 \text{ y } B = 0, X = \overline{0 \cdot 0 + 0 \cdot 1 + 0 \cdot 1} = \overline{0} = 1$$

$$\text{Si } A = 0 \text{ y } C = 0, X = \overline{0 \cdot 1 + 0 \cdot 0 + 1 \cdot 0} = \overline{0} = 1$$

$$\text{Si } B = 0 \text{ y } C = 0, X = \overline{1 \cdot 0 + 1 \cdot 0 + 0 \cdot 0} = \overline{0} = 1$$

5.3 No se puede simplificar.

5.4 No se puede simplificar.

5.5 $X = A + B + C + D$ es válida.

5.6 *Véase* la Figura 5.70.

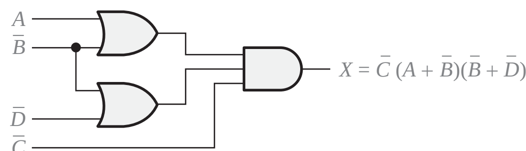


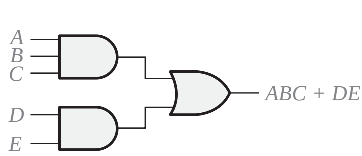
FIGURA 5.70

5.7 $X = \overline{(\overline{\overline{AB}C})(\overline{\overline{DE}F})} = (\overline{AB})C + (\overline{DE})F = (\overline{A} + \overline{B})C + (\overline{D} + \overline{E})F$

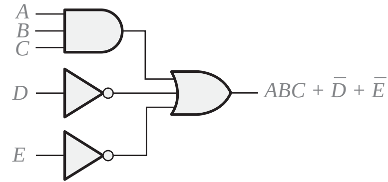
5.8 *Véase* la Figura 5.71.

5.9 $X = \overline{(\overline{\overline{A+B}+C}) + (\overline{\overline{D+E}+F})} = (\overline{A+B}+C)(\overline{D+E}+F) = (\overline{A}\overline{B}+C)(\overline{D}\overline{E}+F)$

5.10 *Véase* la Figura 5.72.



(a)



(b)

FIGURA 5.71

5.11 *Véase* la Figura 5.73.

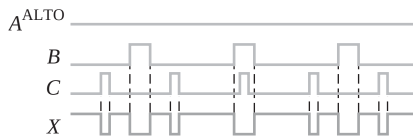


FIGURA 5.72

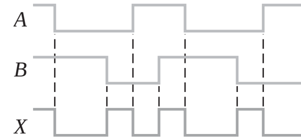


FIGURA 5.73

5.12 *Véase* la Figura 5.74.

5.13 *Véase* la Figura 5.75.

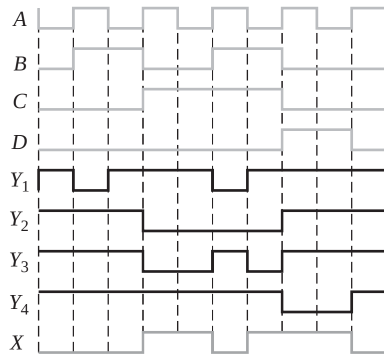


FIGURA 5.74

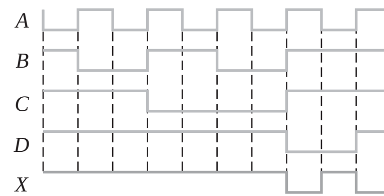


FIGURA 5.75

5.14 G5: NAND_gate2 **port map** (A => IN9, B =>IN10, X = > OUT4);

5.15 *Véase* la Figura 5.76.

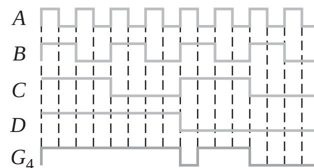


FIGURA 5.76

AUTOTEST

1. (d) **2.** (b) **3.** (c) **4.** (a) **5.** (d) **6.** (b) **7.** (a) **8.** (d)
9. (d) **10.** (e) **11.** (e) **12.** (c)

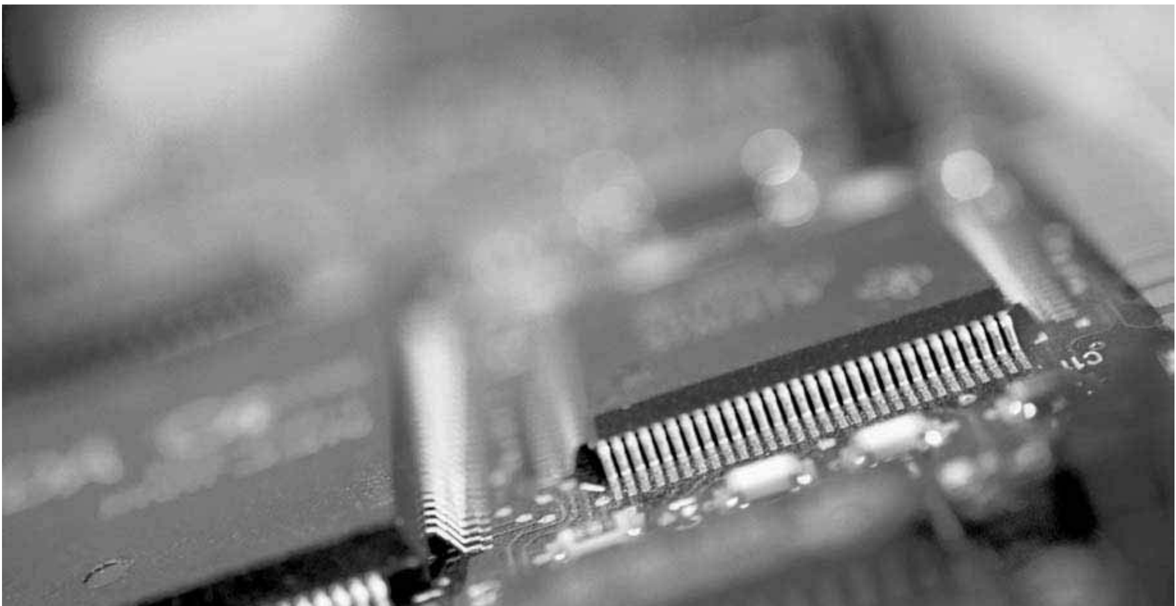
6

FUNCIONES DE LA LÓGICA COMBINACIONAL

CONTENIDO DEL CAPÍTULO

OBJETIVOS DEL CAPÍTULO

- Distinguir entre semi-sumadores y sumadores completos.



- Utilizar sumadores completos para implementar sumadores en paralelo binarios de múltiples bits.
- Explicar las diferencias entre sumadores con acarreo serie y sumadores paralelo con acarreo anticipado.
- Utilizar los comparadores de magnitud para determinar la relación entre dos números binarios y utilizar los comparadores en cascada para realizar comparaciones de números más grandes.
- Implementar un decodificador binario básico.
- Utilizar decodificadores BCD a 7-segmentos en sistemas con displays.
- Emplear un codificador de prioridad BCD-binario en un sencillo sistema con teclado.
- Convertir, utilizando dispositivos lógicos, números en código binario a código Gray, y códigos Gray a números binarios.
- Aplicar multiplexores para la selección de datos, los displays multiplexados, la generación de funciones lógicas y sistemas sencillos de comunicaciones.
- Utilizar decodificadores como demultiplexores.
- Explicar el significado de paridad.
- Usar generadores y comprobadores de paridad para detectar errores de bits en los sistemas digitales.
- Implementar un sencillo sistema de comunicación de datos.
- Identificar *glitches* (impulsos no deseados de muy corta duración), errores muy comunes en los sistemas digitales.

PALABRAS CLAVE

- Semi-sumador
- Conexión en cascada
- Acarreo serie
- Acarreo anticipado
- Decodificador
- Codificador
- Codificador de prioridad
- Multiplexor
- Demultiplexor
- Bit de paridad
- *Glitch*

INTRODUCCIÓN

En este capítulo se presentan distintos tipos de circuitos lógicos combinacionales, incluyendo sumadores, comparadores, decodificadores, codificadores, convertidores de código, multiplexores (selectores de datos), demultiplexores y generadores/comprobadores de paridad. También se incluyen ejemplos de circuitos integrados de función fija.

DISPOSITIVOS LÓGICOS DE FUNCIÓN FIJA

74XX42	74XX47	74XX85
74XX138	74XX139	74XX147
74XX148	74XX151	74XX154
74XX157	74XX280	74XX283

■ ■ ■ APLICACIÓN A LOS SISTEMAS DIGITALES

Esta aplicación a un sistema digital ilustra los conceptos que se enseñan en este capítulo y consiste en una parte de un sistema de control de semáforos. Las secciones “Aplicación a los sistemas digitales” de los Capítulos 6, 7 y 8 se centran en las distintas partes de este mismo sistema. Básicamente, este sistema se encarga de controlar el tráfico en la intersección de una calle con mucho tráfico y otra más despejada. El sistema está formado por una parte en la que se aplica la lógica combinacional, explicada en este capítulo, un circuito de temporización, que se verá en el Capítulo 7 y un sistema de lógica secuencial, al que se aplican los conceptos del Capítulo 8.

6.1 SUMADORES BÁSICOS

Los sumadores son muy importantes no solamente en las computadoras, sino en muchos tipos de sistemas digitales en los que se procesan datos numéricos. Comprender el funcionamiento de un sumador básico es fundamental en el estudio de los sistemas digitales. En esta sección se presentan el semi-sumador y el sumador completo.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Describir el funcionamiento de un semi-sumador. ■ Dibujar el diagrama lógico de un semi-sumador. ■ Describir el funcionamiento de un sumador completo. ■ Dibujar el diagrama lógico de un sumador completo utilizando semi-sumadores. ■ Implementar un sumador completo mediante lógica AND-OR.

El semi-sumador

Recordemos las reglas básicas de la suma binaria expuestas en el Capítulo 2:

0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 10

▲ *Un semi-sumador suma dos bits y genera una suma y una salida de acarreo.*

Todas estas operaciones se realizan mediante un circuito lógico denominado ***semi-sumador***.

Un semi-sumador admite dos dígitos binarios en sus entradas y genera dos dígitos binarios en sus salidas: un bit de suma y un bit de acarreo.

Los semi-sumadores se representan mediante el símbolo lógico de la Figura 6.1.

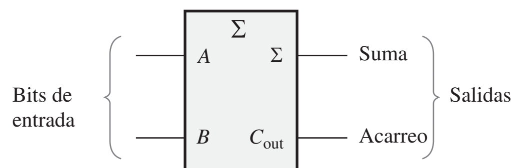


FIGURA 6.1 Símbolo lógico de un semi-sumador.

Lógica del semi-sumador. A partir del funcionamiento lógico de un semi-sumador, expuesto en la Tabla 6.1, las expresiones correspondientes a la suma y al acarreo de salida se pueden obtener como funciones de las entradas. Observe que la salida de acarreo (C_{out}) es 1 sólo cuando A y B son 1; por tanto, C_{out} puede expresarse como una operación AND de las variables de entrada.

Ecuación 6.1 $$C_{out} = AB$$

Observe ahora que la salida correspondiente a la suma (Σ) es 1 sólo si las variables A y B son distintas. Por tanto, la suma puede expresarse como una operación OR-exclusiva de las variables de entrada.

Ecuación 6.2 $$\Sigma = A \oplus B$$

A partir de las Ecuaciones (6.1) y (6.2), se puede desarrollar la implementación lógica del funcionamiento de un semi-sumador. La salida de acarreo se produce mediante una puerta AND, siendo A y B sus dos entra-

A	B	C_{OUT}	Σ
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Σ = suma
C_{out} = acarreo de salida
A y B = variables de entrada (operandos)

TABLA 6.1 Tabla de verdad de un semi-sumador.

das, y la salida de la suma se obtiene mediante una puerta OR-exclusiva, como muestra la Figura 6.2. Recuerde que la operación OR-exclusiva se implementa con puertas AND, una puerta OR e inversores.

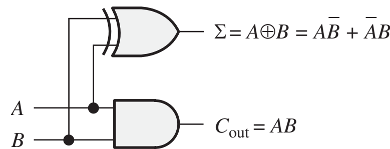


FIGURA 6.2 Diagrama lógico de un semi-sumador.

El sumador completo

▲ *Un sumador completo tiene un acarreo de entrada mientras que el semi-sumador no.*

El segundo tipo de sumador es el ***sumador completo*.**

Un sumador acepta dos bits de entrada y un acarreo de entrada, y genera una salida de suma y un acarreo de salida.

La diferencia principal entre un sumador completo y un semi-sumador es que el sumador completo acepta un acarreo de entrada. El símbolo lógico de un sumador completo se muestra en la Figura 6.3, y la tabla de verdad mostrada en la Tabla 6.2 describe su funcionamiento.

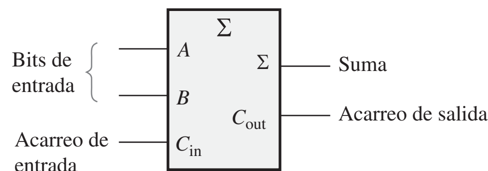


FIGURA 6.3 Símbolo lógico de un sumador completo.

Lógica del sumador completo. El sumador completo tiene que sumar dos bits de entrada y un acarreo de entrada. Del semi-sumador sabemos que la suma de los bits de entrada A y B es la operación OR-exclusiva de esas dos variables, $A \oplus B$. Para sumar el acarreo de entrada (C_{in}) a los bits de entrada, hay que aplicar de nuevo la operación OR-exclusiva, obteniéndose la siguiente ecuación para la salida de suma del sumador completo:

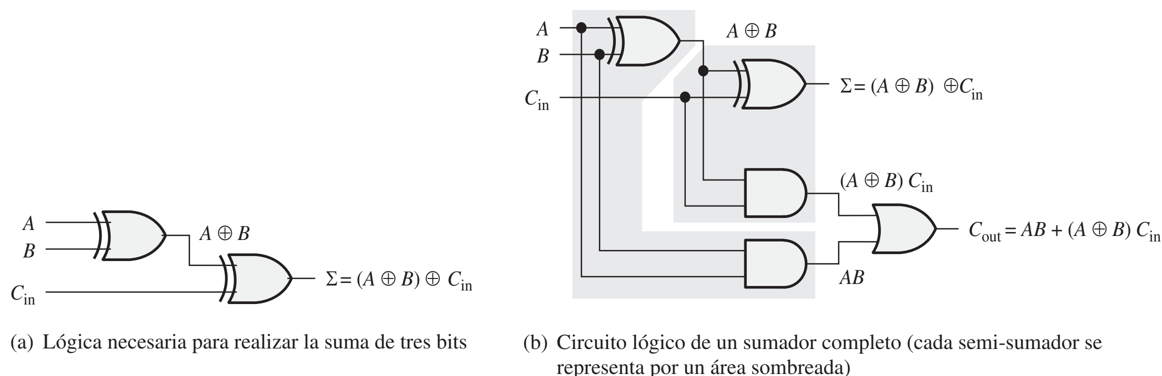
Ecuación 6.3 $$\Sigma = (A \oplus B) \oplus C_{in}$$

A	B	C_{in}	C_{OUT}	Σ
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

C_{in} = acarreo de entrada. Algunas veces se designa como *CI*.
C_{out} = acarreo de salida. Algunas veces se designa como *CO*.
Σ = suma
A y *B* = variables de entrada (operandos)

TABLA 6.2 Tabla de verdad de un sumador completo.

Esto significa que para implementar la función del sumador completo se pueden utilizar dos puertas OR-exclusiva de 2 entradas. La primera tiene que generar el término $A \oplus B$, y la segunda tiene como entradas la salida de la primera puerta XOR y el acarreo de entrada, como se ilustra en la Figura 6.4(a).



(a) Lógica necesaria para realizar la suma de tres bits

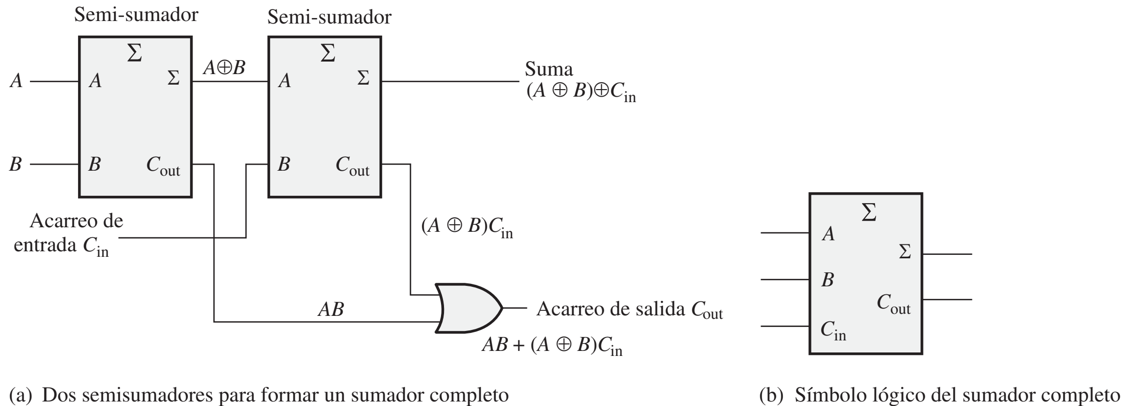
(b) Circuito lógico de un sumador completo (cada semi-sumador se representa por un área sombreada)

FIGURA 6.4 Lógica de un sumador completo.

El acarreo de salida es 1 cuando las dos entradas de la primera puerta XOR están a 1 o cuando las dos entradas de la segunda puerta XOR están a 1. Puede comprobar esto estudiando la Tabla 6.2. El acarreo de salida del sumador completo se obtiene por tanto del producto lógico (AND) de la entradas *A* y *B*, y del producto lógico de $A \oplus B$ y C_{in}. Después se aplica la operación OR a estos dos términos, como muestra la Ecuación 6.4. Esta función se implementa y se combina con la lógica de la suma para formar un circuito sumador completo, como se muestra en la Figura 6.4(b).

Ecuación 6.4 $$C_{out} = AB + (A \oplus B)\, C_{in}$$

Observe que, en la Figura 6.4(b), existen dos semi-sumadores conectados, como se muestra en el diagrama de bloques de la Figura 6.5(a), cuyos acarreos de salida se aplican a una puerta OR. El símbolo lógico mostrado en la Figura 6.5(b) será el que normalmente empleemos para representar un sumador completo.



(a) Dos semisumadores para formar un sumador completo (b) Símbolo lógico del sumador completo

FIGURA 6.5 Sumador completo implementado mediante semi-sumadores.

EJEMPLO 6.1

Para cada uno de los tres sumadores completos de la Figura 6.6, determinar las salidas para las entradas indicadas.

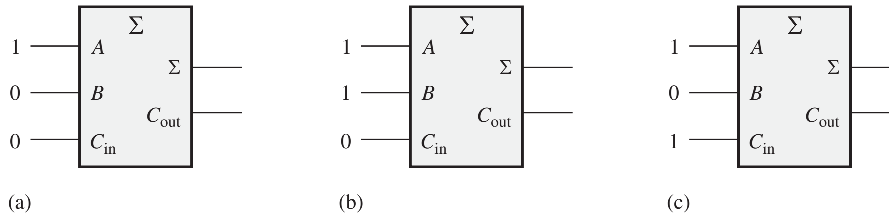


FIGURA 6.6

Solución **(a)** Los bits de entrada son $A= 1$, $B= 0$ y $C_{in}= 0$.

$$1 + 0 + 0 = 1 \text{ sin acarreo}$$

Por tanto, **Σ = 1** y $\boldsymbol{C_{out}= 0}$

(b) Los bits de entrada son $A = 1$, $B = 1$ y $C_{in} = 0$.

$$1 + 1 + 0 = 0 \text{ con acarreo de } 1$$

Por tanto, **Σ = 0** y $\boldsymbol{C_{out} = 1}$

(c) Los bits de entrada son $A= 1$, $B = 0$ y $C_{in} = 1$.

$$1 + 0 + 1 = 0 \text{ con acarreo de } 1$$

Por tanto, **Σ = 0** y $\boldsymbol{C_{out}= 1}$

Problema relacionado* ¿Cuáles serán las salidas del sumador completo para $A = 1$, $B = 1$ y $C_{in}= 1$?

* Las respuestas se encuentran al final del capítulo.

REVISIÓN DE LA SECCIÓN 6.1

1. Determinar la suma (Σ) y el acarreo de salida (C_{out}) de un semi-sumador para cada uno de los siguientes grupos de bits de entrada:
(a) 01 **(b)** 00 **(c)** 10 **(d)** 11
2. Un sumador completo tiene C_{in} = 1. ¿Cuánto vale la suma (Σ) y el acarreo de salida (C_{out}) cuando A = 1 y B = 1?

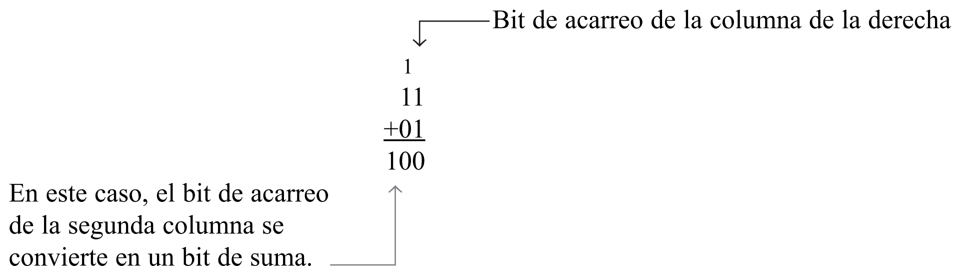
6.2 SUMADORES BINARIOS EN PARALELO

Para formar un sumador binario en paralelo se conectan dos o más sumadores completos. En esta sección aprenderemos los principios básicos de este tipo de sumador, de manera que podamos entender todas las funciones necesarias de entrada y salida cuando se trabaja con este tipo de dispositivos.

Al finalizar esta sección, el lector deberá ser capaz de:

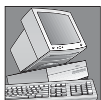
■ Utilizar sumadores completos para implementar un sumador binario en paralelo. ■ Explicar el proceso de adición en un sumador binario en paralelo. ■ Emplear la tabla de verdad para un sumador en paralelo de 4 bits. ■ Utilizar dos dispositivos 74LS283 para sumar dos números binarios de 4 bits. ■ Ampliar el sumador de 4 bits para poder realizar adiciones de 8 bits o 16 bits.

Como se ha visto en la Sección 6.1, un único sumador completo es capaz de sumar dos números binarios de 1 bit y un acarreo de entrada. Para sumar números binarios de más de un bit, se tienen que utilizar sumadores completos adicionales. Cuando se suman dos números binarios, cada columna genera un bit de suma y un 1 ó 0, correspondiente al bit de acarreo, que se añade a la columna inmediata de la izquierda, como se muestra a continuación con dos números de 2 bits.



Para sumar dos números binarios, se necesita un sumador completo por cada bit que tengan los números que se quieren sumar. Así, para números de dos bits se necesitan dos sumadores, para números de cuatro bits hacen falta cuatro sumadores, y así sucesivamente. La salida de acarreo de cada sumador se conecta a la entrada de acarreo del sumador de orden inmediatamente superior, como se muestra en la Figura 6.7 para un sumador de 2 bits. Téngase en cuenta que se puede usar un semi-sumador para la posición menos significativa, o bien se puede poner a 0 (masa) la entrada de acarreo de un sumador completo, ya que no existe entrada de acarreo en la posición del bit menos significativo.

En la Figura 6.7 los bits menos significativos (LSB) de los dos números se representan como A_1 y B_1. Los siguientes bits de orden superior se representan como A_2 y B_2. Los tres bits de suma son Σ_1, Σ_2 y Σ_3. Observe



NOTAS INFORMÁTICAS

Las computadoras realizan la operación de suma con dos números a un tiempo, denominados *operandos*. El *operando fuente* es un número que se añade a un número existente denominado *operando de destino*, que es el que se almacena en un registro de la UAL, tal como el acumulador. A continuación, la suma de los dos números se almacena de nuevo en el acumulador. La adición se realiza con números enteros o números en coma flotante utilizando, respectivamente, las instrucciones ADD o FADD.

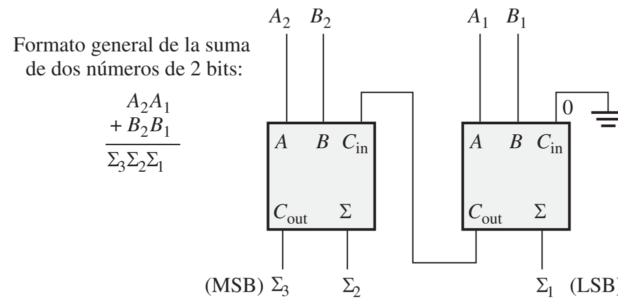


FIGURA 6.7 Diagrama de bloques de un sumador paralelo de 2 bits básico utilizando dos sumadores completos.

que el acarreo de salida del sumador completo de más a la izquierda se convierte en el bit más significativo (MSB) en la suma Σ_3.

EJEMPLO 6.2

Determinar la suma generada por el sumador paralelo de tres bits mostrado en la Figura 6.8 e indicar los acarreos intermedios cuando se están sumando los números 101 y 011.

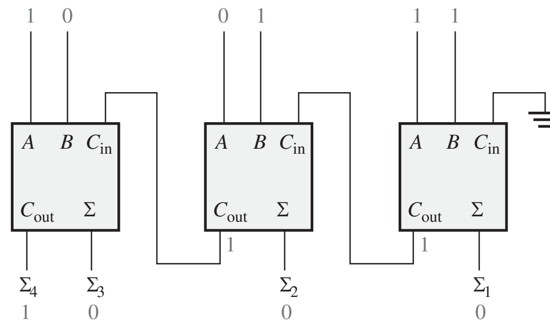


FIGURA 6.8

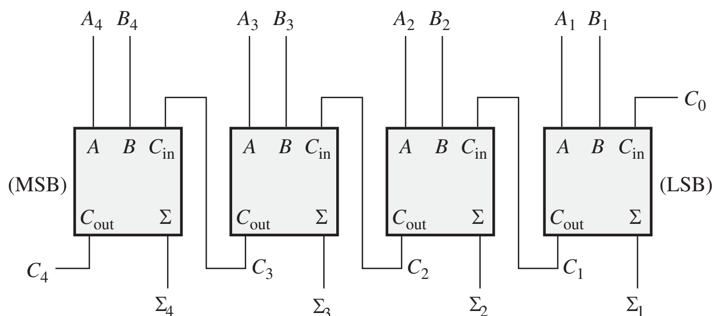
Solución Los bits menos significativos (LSB) de los dos números se suman en el sumador completo situado más a la derecha. En la Figura 6.8 se indican los bits de suma y los acarreos intermedios en negrita.

Problema relacionado Cuando se utiliza un sumador paralelo de 3 bits para sumar los números 111 y 101, ¿cuáles son las salidas de suma?

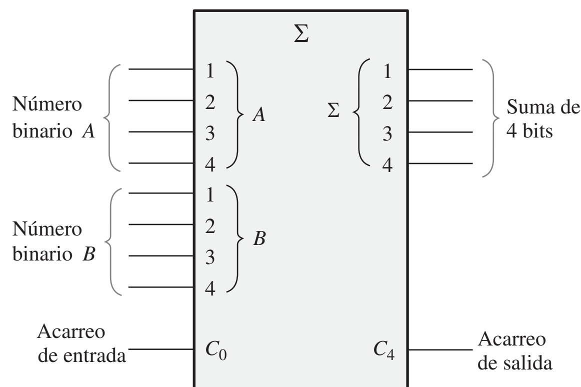
Sumadores en paralelo de cuatro bits

Un grupo de cuatro bits se denomina **nibble**. Un sumador básico en paralelo de 4 bits se implementa mediante cuatro sumadores completos, como se muestra en la Figura 6.9. De nuevo, los bits menos significativos (A_1 y B_1) de cada número que se suma, se introducen en el sumador completo que está más a la derecha; los bits

de orden más alto se introducen sucesivamente en los siguientes sumadores, aplicando los bits más significativos de cada número (A_4 y B_4) al sumador que está más a la izquierda. La salida de acarreo de cada sumador se conecta a la entrada de acarreo del siguiente sumador de orden superior. Estos acarreos se denominan *acarreos internos*.



(a) Diagrama de bloques



(b) Símbolo lógico

FIGURA 6.9 Sumador en paralelo de 4 bits.

En la mayoría de las hojas de características suministradas por los fabricantes, se denomina C_0 al acarreo de entrada del sumador del bit menos significativo; C_4, en el caso de cuatro bits, sería el acarreo de salida del sumador del bit más significativo; Σ_1 (LSB) hasta Σ_4 (MSB) son las sumas de salida. El símbolo lógico correspondiente se muestra en la Figura 6.9(b).

En función del método utilizado para manipular los acarreos en un sumador paralelo, existen dos tipos: el sumador de *acarreo serie* y el sumador *de acarreo anticipado*, que se estudian en la Sección 6.3.

Tabla de verdad de un sumador en paralelo de 4 bits

La Tabla 6.3 es la tabla de verdad de un sumador de 4 bits. En algunas hojas de características, las tablas de verdad se denominan *tablas de función* o *tablas de verdad funcionales*. El subíndice *n* representa los bits del sumador y puede ser igual a 1, 2, 3 o 4 para un sumador de 4 bits. C_{n-1} es el acarreo del sumador previo. Los acarreos C_1, C_2 y C_3 se generan internamente. C_0 es un acarreo de entrada externo y C_4 es una salida. El Ejemplo 6.3 ilustra cómo utilizar la Tabla 6.3.

C_{n-1}	A_n	B_n	Σ_n	C_n
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

TABLA 6.3 Tabla de verdad para cada etapa de un sumador en paralelo de 4 bits.

EJEMPLO 6.3

Utilizar la tabla de verdad del sumador en paralelo de 4 bits (Tabla 6.3) para hallar la suma y el acarreo de salida correspondientes a los siguientes dos números binarios de 4 bits, siendo el acarreo de entrada (C_{n-1}) igual a 0:

$$A_4A_3A_2A_1 = 1100 \quad \text{y} \quad B_4B_3B_2B_1 = 1100$$

Solución Para $n = 1$: $A_1 = 0$, $B_1 = 0$ y $C_{n-1} = 0$. Según la primera fila de la tabla,

$$\mathbf{\Sigma_1 = 0} \text{ y } C_1 = 0$$

Para $n = 2$: $A_2 = 0$, $B_2 = 0$ y $C_{n-1} = 0$. Según la primera fila de la tabla,

$$\mathbf{\Sigma_2 = 0} \text{ y } C_2 = 0$$

Para $n = 3$: $A_3 = 1$, $B_3 = 1$ y $C_{n-1} = 0$. Según la cuarta fila de la tabla,

$$\mathbf{\Sigma_3 = 0} \text{ y } C_3 = 1$$

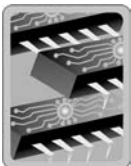
Para $n = 4$: $A_4 = 1$, $B_4 = 1$ y $C_{n-1} = 1$. Según la última fila de la tabla,

$$\mathbf{\Sigma_4 = 0} \text{ y } C_4 = \mathbf{1}$$

C_4 será el acarreo de salida; la suma de 1100 y 1100 es 11000.

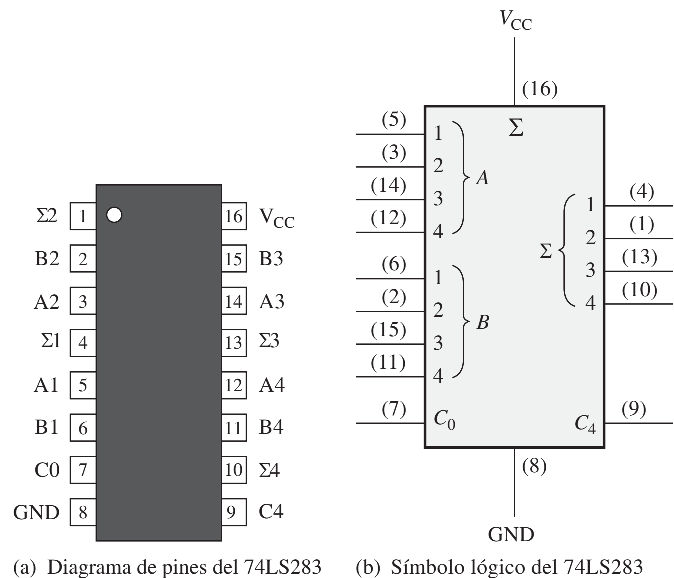
Problema relacionado Utilizar la tabla de verdad (Tabla 6.3) para calcular la suma de los números binarios 1011 y 1010.

SUMADORES PARALELO DE 4 BITS 74LS283



Un ejemplo de un sumador paralelo de 4 bits que está disponible como circuito integrado es el 74LS283. Para el 74LS83, V_{CC} es el pin 16 y tierra es el pin 8, que es una configuración estándar. El diagrama de pines y el símbolo lógico de este dispositivo se muestran en la Figura 6.10. Este dispositivo está disponible en las familias TTL y CMOS. Consulte el sitio web de Texas Instruments en www.ti.com.

Hoja de características del CI. Recuerde que las puertas lógicas tienen un retardo de propagación especificado, t_P, desde una entrada a la salida. Para estos dispositivos lógicos, existen varias especificaciones diferentes para este parámetro. El sumador paralelo de 4 bits dispone de 4 especificaciones para t_P, como se muestra en la Figura 6.11, tabla que es parte de una hoja de características del 74LS283.



(a) Diagrama de pines del 74LS283 (b) Símbolo lógico del 74LS283

FIGURA 6.10 Sumador en paralelo de 4 bits.

Símbolo	Parámetro	Límites			Unidad
		Mínimo	Típico	Máximo	
t_{PLH} t_{PHL}	Retardo de propagación; entrada de acarreo (C_0) a cualquier salida de suma Σ.		16 15	24 24	ns
t_{PLH} t_{PHL}	Retardo de propagación; cualquier entrada A o B a salidas de suma Σ.		15 15	24 24	ns
t_{PLH} t_{PHL}	Retardo de propagación; entrada de acarreo (C_0), a la salida de acarreo (C_4).		11 11	17 22	ns
t_{PLH} t_{PHL}	Retardo de propagación; cualquier entrada A o B a la salida de acarreo C_4.		11 12	17 17	ns

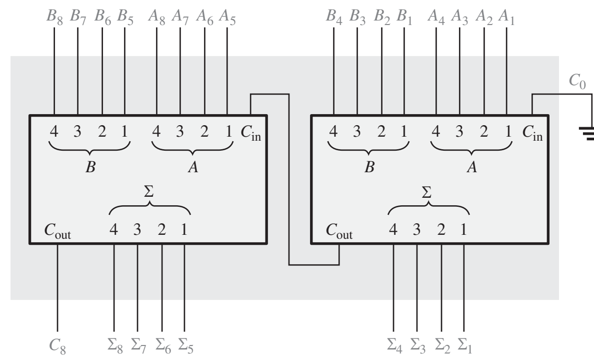
FIGURA 6.11 Características del retardo de propagación del 74LS283.

Expansión de sumadores

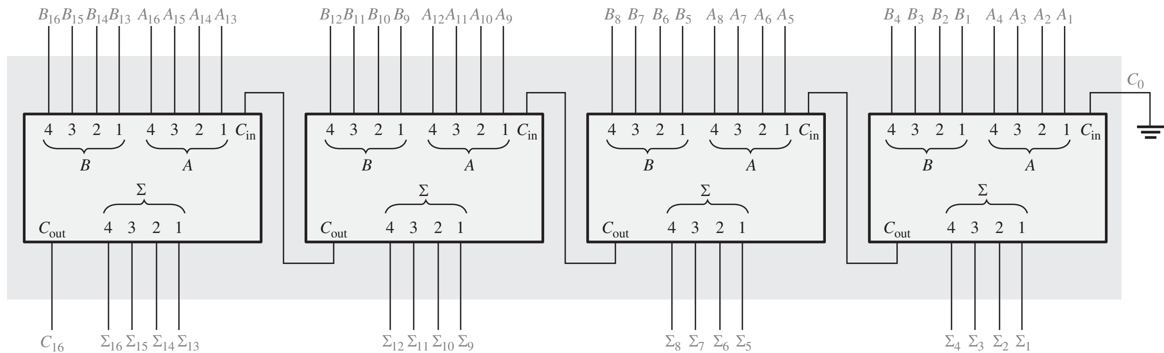
▲ *Los sumadores pueden ampliarse conectándose en cascada para trabajar con más bits.*

Un sumador en paralelo de 4 bits se puede expandir para realizar sumas de dos números de 8 bits, utilizando dos sumadores de cuatro bits. La entrada de acarreo del sumador de menor orden (C_0) se conecta a tierra, ya que no existe acarreo en la posición del bit menos significativo, y la salida de acarreo del sumador de menor orden se conecta a la entrada de acarreo del sumador de orden superior, como se muestra en la Figura 6.12(a). Este proceso se denomina ***conexión en cascada***. Observe que, en este caso, el acarreo de salida se designa como C_8, dado que se genera a partir del bit que se encuentra en la posición número ocho. El sumador de menor orden es el que realiza la suma de los cuatro bits menos sig-

nificativos, mientras que el sumador de orden superior es el que suma los cuatro bits más significativos de los dos números binarios de 8 bits.



(a) Sumadores de 4 bits conectados en cascada que forman un sumador de 8 bits



(b) Sumadores de 4 bits conectados en cascada que forman un sumador de 16 bits

FIGURA 6.12 Ejemplos de expansión de sumadores.

De forma similar, se pueden emplear cuatro sumadores de 4 bits en cascada para sumar dos números de 16 bits, como se muestra en la Figura 6.12(b). Observe que el acarreo de salida se designa como C_{16}, ya que se genera a partir del bit que se encuentra en la posición dieciséis.

EJEMPLO 6.4

Mostrar cómo se pueden conectar dos sumadores 74LS283 para formar un sumador en paralelo de 8 bits. Obtener los bits de salida para los siguientes números de entrada de 8 bits:

$$A_8\,A_7\,A_6\,A_5\,A_4\,A_3\,A_2\,A_1 = 10111001 \quad \text{y} \quad B_8\,B_7\,B_6\,B_5\,B_4\,B_3\,B_2\,B_1 = 10011110$$

Solución Se utilizan dos sumadores paralelo de 4 bits 74LS283 para implementar el sumador de 8 bits. La única conexión entre los dos 74LS283 es la que une la salida de acarreo (pin 9) del sumador de menor orden a la entrada de acarreo (pin 7) del sumador de orden superior, como se muestra en la Figura 6.13. El pin 7 del sumador de orden inferior se conecta a masa (no hay entrada de acarreo).

La suma de los dos números de 8 bits es:

$$\Sigma_9\Sigma_8\Sigma_7\Sigma_6\Sigma_5\Sigma_4\Sigma_3\Sigma_2\Sigma_1 = 101010111$$

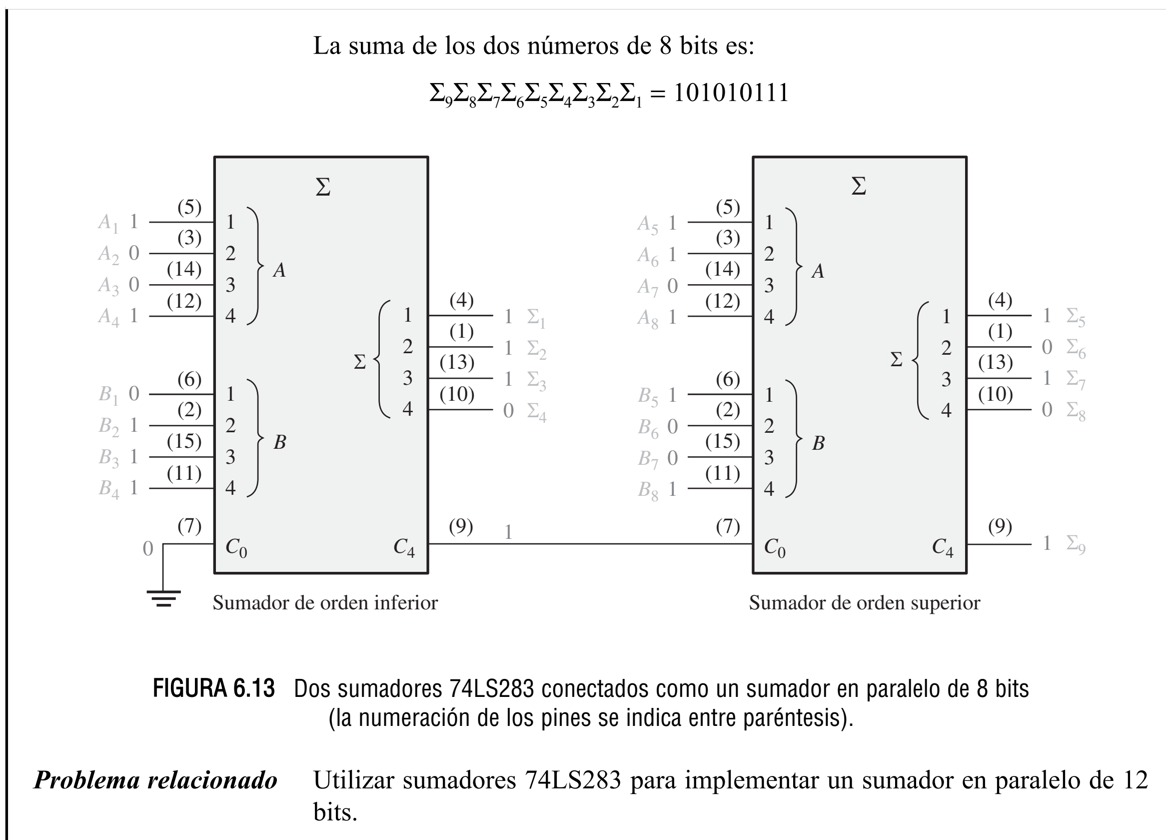


FIGURA 6.13 Dos sumadores 74LS283 conectados como un sumador en paralelo de 8 bits (la numeración de los pines se indica entre paréntesis).

Problema relacionado Utilizar sumadores 74LS283 para implementar un sumador en paralelo de 12 bits.

Aplicación

Un ejemplo de aplicación de un sumador completo y de un sumador en paralelo es un sencillo sistema de recuento de votos, que se puede utilizar simultáneamente para proporcionar el número de votos afirmativos y el de negativos. Por ejemplo, este tipo de sistema podría utilizarse en una reunión de personas donde se necesite determinar de forma inmediata el número de opiniones favorables para tomar decisiones, o votar sobre determinados temas.

En su forma más sencilla, el sistema se compone de un interruptor para seleccionar las dos posibles opciones (afirmativa o negativa) de cada persona en la reunión y un display digital para mostrar el número de votos afirmativos, y otro para los negativos. El sistema básico se muestra en la Figura 6.14, para un número de personas igual a 6, pero se puede ampliar a cualquier número de votantes, añadiendo nuevos módulos de 6 posiciones, sumadores en paralelo y circuitos de displays adicionales.

En la Figura 6.14, cada sumador completo puede generar la suma de hasta tres votos. La suma y el acarreo de salida de cada sumador completo se conectan a las dos entradas de menor orden de un sumador binario en paralelo. Las dos entradas de orden superior del sumador en paralelo se conectan a tierra (0), ya que nunca existe la posibilidad de que la entrada binaria sea mayor que 0011 (3 decimal). Para este sistema básico de 6 posiciones, las salidas del sumador en paralelo se conectan a un decodificador BCD a 7-segmentos que controla el display de 7-segmentos. Como ya se ha mencionado, se tienen que añadir circuitos adicionales si se decide ampliar el sistema.

Las resistencias entre las entradas de cada sumador completo y tierra aseguran que cada entrada se encuentra a nivel BAJO cuando el interruptor está en su posición neutra (se utiliza lógica CMOS). Cuando un inte-

rruptor se mueve a la posición “sí” o a la posición “no”, se aplica un nivel de tensión ALTO (V_{CC}) a la entrada del sumador completo asociado.

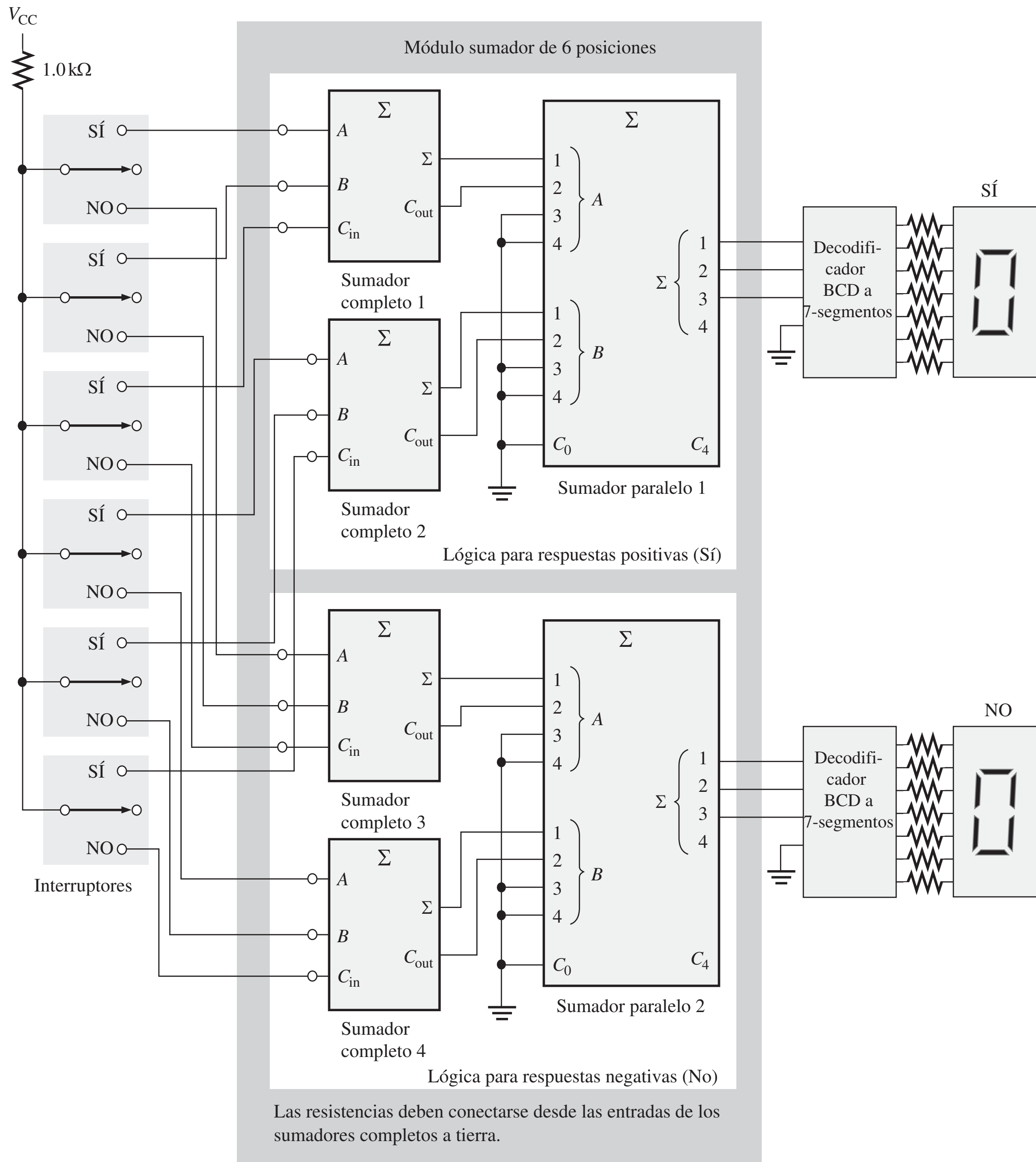


FIGURA 6.14 Sistema de votación realizado a partir de sumadores completos y sumadores binarios en paralelo.

REVISIÓN DE LA SECCIÓN 6.2

1. Se aplican dos números binarios de 4 bits (1101 y 1011) a un sumador en paralelo de 4 bits. El acarreo de entrada es 1. Determinar la suma (Σ) y el acarreo de salida.
2. ¿Cuántos sumadores 74LS283 se necesitarán para sumar dos números binarios, que representan números decimales iguales o inferiores a 1000_{10}?

6.3 SUMADORES DE ACARREO SERIE Y DE ACARREO ANTICIPADO

Como se ha mencionado en la sección anterior, los sumadores paralelo pueden clasificarse en dos categorías dependiendo de la forma en que se manejan los acarreos internos de una etapa a otra. Estas categorías son: acarreo serie y acarreo anticipado. Externamente, ambos tipos de sumador son iguales en términos de entradas y salidas. La diferencia se encuentra en la velocidad a la que se suman los números. El sumador de acarreo anticipado es mucho más rápido que el sumador de acarreo serie.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Explicar la diferencia entre un sumador de acarreo anticipado y un sumador de acarreo serie. ■ Explicar las ventajas de la operación suma utilizando acarreo anticipado. ■ Definir *generación de acarreo* y *propagación de acarreo*, y explicar la diferencia. ■ Desarrollar lógica de acarreo anticipado. ■ Explicar por qué los CI 74LS283 conectados en cascada presentan propiedades de acarreo en serie y de acarreo anticipado.

Sumador de acarreo serie

Un sumador de ***acarreo serie*** es aquel en el que la salida de acarreo de cada sumador completo se conecta a la entrada de acarreo de la siguiente etapa de orden inmediatamente superior (una etapa es un sumador completo). La suma y el acarreo de salida de cualquier etapa no se pueden generar hasta que tiene lugar el acarreo de entrada, lo que da lugar a un retardo temporal en el proceso de adición, como se muestra en la Figura 6.15. El retardo de propagación del acarreo para cada sumador completo es el tiempo transcurrido desde la apli-

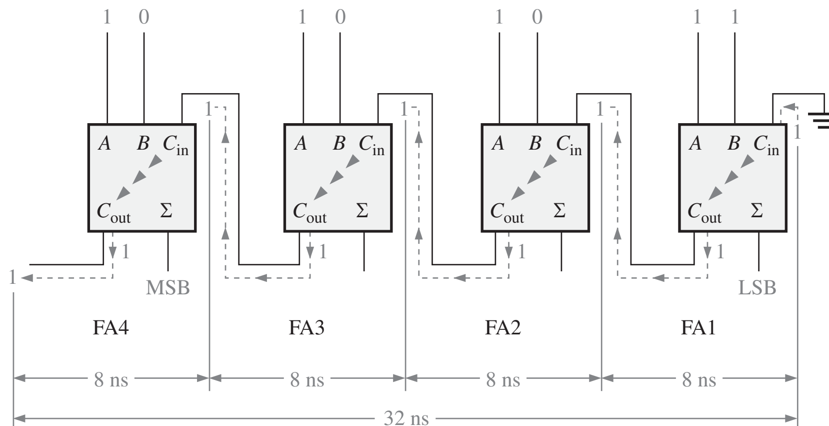


FIGURA 6.15 Un sumador paralelo de 4 bits con acarreo serie, que muestra los retardos de propagación del acarreo del caso peor.

cación del acarreo de entrada hasta que se produce el acarreo de salida, suponiendo que las entradas *A* y *B* ya existan.

El sumador completo 1 (FA1) no puede generar un acarreo de salida hasta que se aplique un acarreo de entrada. El sumador completo 2 (FA2) no puede generar un potencial acarreo de salida hasta que el sumador completo 1 produzca un acarreo de salida. El sumador completo 3 (FA3) no puede generar un potencial acarreo de salida hasta que FA1 produzca un acarreo de salida seguido de un acarreo de salida de FA2, y así sucesivamente. Como se puede ver en la Figura 6.15, el acarreo de entrada de la etapa menos significativa se transmite en serie a través de todos los sumadores antes de se produzca una suma final. El retardo acumulado a través de todas las etapas de sumador es el tiempo de suma del "caso peor". El retardo total puede variar dependiendo del bit de acarreo generado por cada sumador completo. Si se suman dos números que no generan acarreos (0) entre las etapas, el tiempo de suma es simplemente el tiempo de propagación de un solo sumador desde que se aplican los bits de datos en las entradas hasta que aparece la salida de suma.

Sumador de acarreo anticipado

La velocidad a la que se puede efectuar una suma está limitada por el tiempo necesario para que se propaguen los acarreos a través de todas las etapas de un sumador paralelo. Un método que permite acelerar el proceso de adición eliminando este retardo del acarreo serie es la adición con ***acarreo anticipado***. El sumador con acarreo anticipado anticipa el acarreo de salida de cada etapa y, en función de los bits de entrada de cada etapa, genera el acarreo de salida bien mediante la generación de acarreo o la propagación de acarreo.

La **generación de acarreo** tiene lugar cuando el sumador completo genera internamente un acarreo de salida. Sólo cuando ambos bits de entrada son 1 se genera un acarreo. El acarreo generado, C_g, se expresa como la función AND de los 2 bits de entrada, *A* y *B*.

Ecuación 6.5 $C_g = AB$

La **propagación de acarreo** tiene lugar cuando el acarreo de entrada se transmite como acarreo de salida. Un acarreo de entrada puede ser propagado por el sumador completo cuando uno o ambos bits de entrada son igual a 1. El acarreo propagado, C_p, se expresa como la función OR de los bits de entrada.

Ecuación 6.6 $C_p = A + B$

En la Figura 6.16 se ilustran las condiciones para la generación de acarreo y la propagación de acarreo. Las tres puntas de flecha simbolizan la propagación.

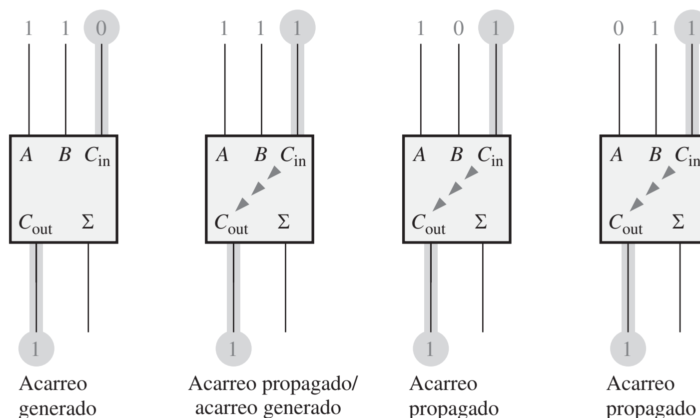


FIGURA 6.16 Ilustración de las condiciones de la generación y la propagación de acarreo.

El acarreo de salida de un sumador completo puede expresarse en función del acarreo generado (C_g) y el acarreo propagado (C_p). El acarreo de salida (C_{out}) es un 1 si el acarreo generado es 1 O si el acarreo propagado es 1 Y el acarreo de entrada (C_{in}) es 1. En otras palabras, obtenemos un acarreo de salida de 1 si el sumador completo los genera (A = 1 AND B = 1) o si el sumador propaga el acarreo de entrada (A = 1 OR B = 1) AND C_{in} = 1. Esta relación se expresa del siguiente modo:

Ecuación 6.7 $C_{out} = C_g + C_pC_{in}$

Veamos ahora cómo se puede aplicar este concepto a un sumador paralelo, cuyas etapas individuales se muestran en el ejemplo de 4 bits de la Figura 6.17. Para sumador completo, el acarreo de salida depende del acarreo generado (C_g), el acarreo propagado (C_p) y su acarreo de entrada (C_{in}). Las funciones C_g y C_p para cada etapa están disponibles *de forma inmediata* tan pronto como se aplican los bits de entrada A y B y el acarreo de entrada del sumador menos significativo (LSB), ya que sólo dependen de estos bits. El acarreo de entrada de cada etapa es el acarreo de salida de la etapa anterior.

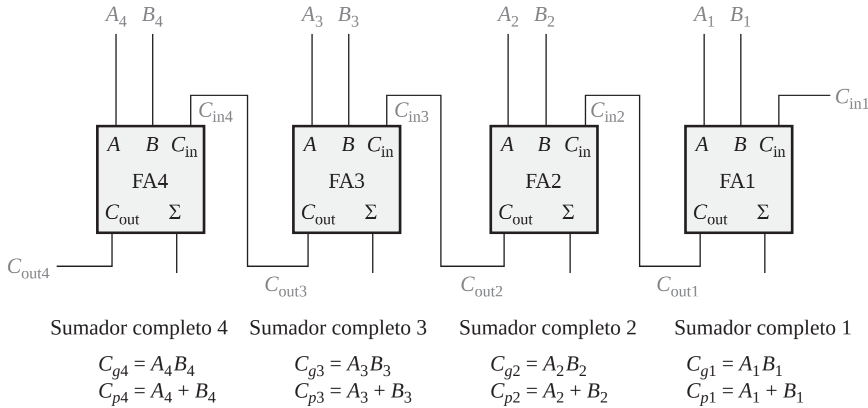


FIGURA 6.17 Generación de acarreo y propagación de acarreo en función de los bits de entrada en un sumador de 4 bits.

Basándonos en este análisis, podemos desarrollar expresiones para el acarreo de salida, C_{out} de cada etapa de sumador completo para el ejemplo de 4 bits.

Sumador completo 1:

$$C_{out1} = C_{g1} + C_{p1}C_{in1}$$

Sumador completo 2:

$$C_{in2} = C_{out1}$$

$$C_{out2} = C_{g2} + C_{p2}C_{in2} = C_{g2} + C_{p2}C_{out1} = C_{g2} + C_{p2}(C_{g1} + C_{p1}C_{in1})$$

$$= C_{g2} + C_{p2}C_{g1} + C_{p2}C_{p1}C_{in1}$$

Sumador completo 3:

$$C_{in3} = C_{out2}$$

$$C_{out3} = C_{g3} + C_{p3}C_{in3} = C_{g3} + C_{p3}C_{out2} = C_{g3} + C_{p3}(C_{g2} + C_{p2}C_{g1} + C_{p2}C_{p1}C_{in1})$$

$$= C_{g3} + C_{p3}C_{g2} + C_{p3}C_{p2}C_{g1} + C_{p3}C_{p2}C_{p1}C_{in1}$$

Sumador completo 4:

$$C_{in4} = C_{out3}$$
$$C_{out4} = C_{g4} + C_{p4}C_{in4} = C_{g4} + C_{p4}C_{out3}$$
$$= C_{g4} + C_{p4}(C_{g3} + C_{p3}C_{g2} + C_{p3}C_{p2}C_{g1} + C_{p3}C_{p2}C_{p1}C_{in1})$$
$$= C_{g4} + C_{p4}C_{g3} + C_{p4}C_{p3}C_{g2} + C_{p4}C_{p3}C_{p2}C_{g1} + C_{p4}C_{p3}C_{p2}C_{p1}C_{in1}$$

Observe que en cada una de estas expresiones, el acarreo de salida para cada etapa de sumador completo sólo depende del acarreo de entrada inicial (C_{in1}), las funciones C_g y C_p de dicha etapa y las funciones C_g y C_p de las etapas anteriores. Puesto que cada una de las expresiones de C_g y C_p pueden expresarse en función de las entradas A y B a los sumadores completos, todos los acarreos de salida están inmediatamente disponibles (excepto por los retardos de puerta) y no es necesario esperar a que se propague un acarreo a través de todas la etapas antes de obtener el resultado final. Por tanto, la técnica del acarreo anticipado acelera el proceso de adición.

Las ecuaciones de C_{out} se implementan con puertas lógicas y se conectan a los sumadores completos para crear un sumador de 4 bits con acarreo anticipado, como el que se muestra en la Figura 6.18.

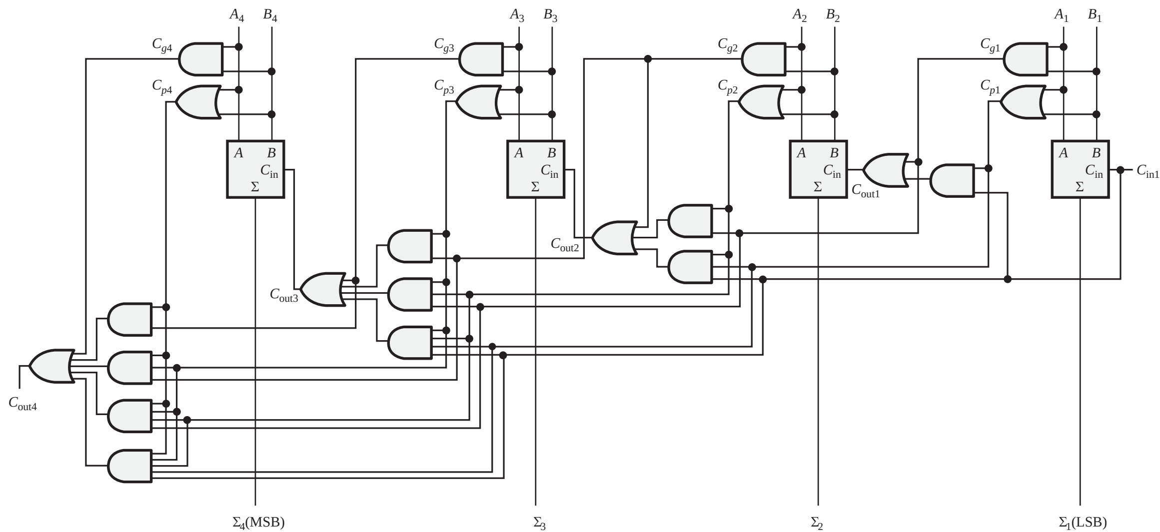


FIGURA 6.18 Diagrama lógico de un sumador de 4 etapas con acarreo anticipado.

Combinación de sumadores de acarreo serie y acarreo anticipado

En la Sección 6.2 se ha presentado el sumador de 4 bits 74LS283, que es un sumador con acarreo anticipado. Cuando estos sumadores se conectan en cascada para ampliar su capacidad de manejar números binarios de más de cuatro bits, el acarreo de salida de un sumador se conecta al acarreo de entrada del siguiente. Esto crea una condición de acarreo serie entre los sumadores de 4 bits, de modo que cuando se conectan en cascada dos o más 74LS283, el sumador resultante realmente es una combinación de un sumador con acarreo serie y acarreo anticipado. La operación de acarreo anticipado es interna en cada sumador MSI y la función de acarreo serie entra en juego cuando un acarreo pasa de un sumador al siguiente.

REVISIÓN DE LA SECCIÓN 6.3

1. Los bits de entrada de un sumador completo son $A = 1$ y $B = 0$. Determinar C_g y C_p.
2. Determinar el acarreo de salida de un sumador completo cuando $C_{in} = 1$, $C_g = 0$ y $C_p = 1$.

6.4 COMPARADORES

La función básica de un comparador consiste en comparar las magnitudes de dos cantidades binarias para determinar su relación. En su forma más sencilla, un circuito comparador determina si dos números son iguales.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Utilizar una puerta OR-exclusiva como comparador básico. ■ Analizar la lógica interna de un comparador de magnitud que posee tanto salida de igualdad como de desigualdad. ■ Utilizar el comparador 74HC85 para la comparación de dos números binarios de 4 bits. ■ Conectar en cascada comparadores 74HC85, para comparar números de ocho o más bits.

Igualdad

Como ya vimos en el Capítulo 3, la puerta OR-exclusiva se puede emplear como un comparador básico, ya que su salida es 1 si sus dos bits de entrada son diferentes y 0 si son iguales. La Figura 6.19 muestra una puerta OR-exclusiva utilizada como comparador de 2 bits.

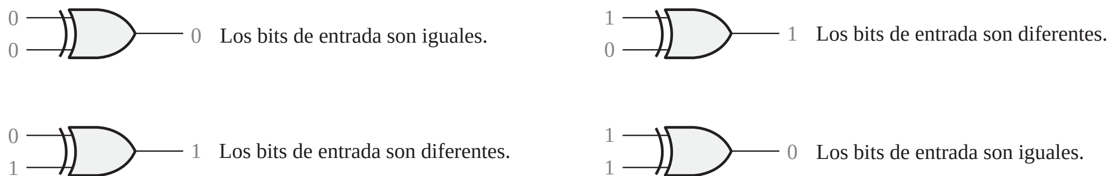


FIGURA 6.19 Funcionamiento del comparador básico.

Para comparar números binarios de dos bits, se necesita una puerta OR-exclusiva adicional. Los dos bits menos significativos (LSB) de ambos números se comparan mediante la puerta G_1 y los dos más significativos (MSB) son comparados mediante la puerta G_2, como se muestra en la Figura 6.20. Si los dos números son iguales, sus correspondientes bits también lo son, y la salida de cada puerta OR-exclusiva será 0. Si los correspondientes conjuntos de bits no son idénticos, la salida de la puerta OR-exclusiva será un 1.

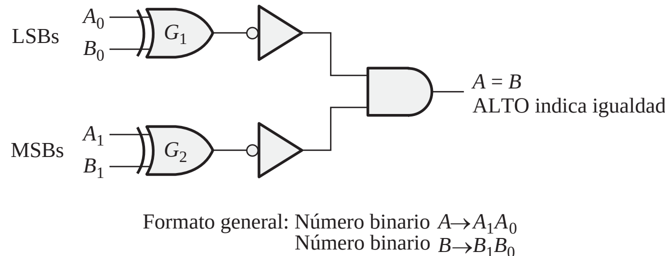


FIGURA 6.20 Diagrama lógico de la comparación de igualdad de dos números de 2 bits.

▲ *Un comparador determina si dos números binarios son iguales o distintos.*

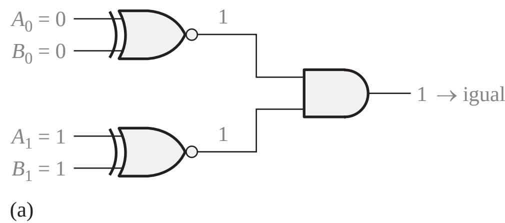
Para obtener un único resultado de salida que indique la igualdad o desigualdad entre los dos números, se pueden usar dos inversores y una puerta AND, como muestra la Figura 6.20. La salida de cada puerta OR-exclusiva se invierte y se aplica a la entrada de la puerta AND. Cuando los bits de entrada de cada OR-exclusiva son iguales, lo que quiere decir que los bits de ambos números son iguales, las entradas de la puerta AND son 1, por lo que el resultado a su salida también será 1. Cuando los dos números no son iguales, al menos uno o ambos conjuntos de bits será distinto, lo que da lugar a, al menos, un 0 en una de las entradas de la puerta AND, y el resultado a su salida será 0. Por tanto, la salida de la puerta AND indica la igualdad (1) o desigualdad (0) entre dos números.

El Ejemplo 6.5 ilustra esta operación para dos casos específicos. La puerta OR-exclusiva y el inversor se han reemplazado por un símbolo NOR-exclusiva.

EJEMPLO 6.5

Aplicar cada uno de los siguientes conjuntos de números binarios a las entradas del comparador de la Figura 6.21 y determinar la salida, evaluando los niveles lógicos a través del circuito.

(a) 10 y 10 **(b)** 11 y 10



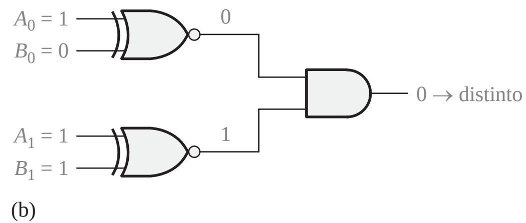


FIGURA 6.21

Solución **(a)** La salida es **1** para las entradas 10 y 10, como se muestra en la Figura 6.21(a).

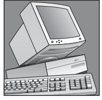
(b) La salida es **0** para las entradas 11 y 10, como se muestra en la Figura 6.21(b).

Problema relacionado Repetir el proceso para las entradas binarias 01 y 10.

Como ya se ha visto en el Capítulo 3, un circuito comparador básico se puede ampliar para poder tratar cualquier número de bits. La puerta AND establece la condición de que todos los bits de los dos números que se comparan tienen que ser iguales si los números lo son.

Desigualdad

Además de disponer de una salida que indica si los dos números son iguales, muchos circuitos integrados comparadores tienen salidas adicionales que indican cuál de los dos números que se comparan es el mayor. Esto significa que existe una salida que indica cuándo el número A es mayor que el número B ($A > B$) y otra salida que indica cuándo A es menor que B ($A < B$), como se muestra en el símbolo lógico del comparador de cuatro bits de la Figura 6.22.



NOTAS INFORMÁTICAS

En una computadora, la *caché* es una memoria intermedia muy rápida entre la unidad de procesamiento central (CPU) y la memoria principal, más lenta. La CPU solicita los datos enviando la *dirección* (ubicación unívoca) en memoria. Parte de esta dirección se denomina *marcador*. El *comparador de marcadores de dirección* compara el marcador de la CPU con el marcador del directorio de la caché. Si ambos son iguales, quiere decir que los datos direccionados se encuentran ya en la caché y se recuperan de forma muy rápida. Si los marcadores son distintos, los datos deben recuperarse de la memoria principal a una velocidad mucho más lenta.

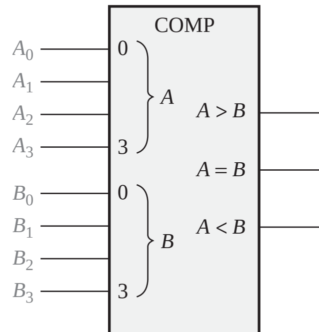


FIGURA 6.22 Símbolo lógico para un comparador de 4 bits con indicación de desigualdad.

Para determinar una desigualdad entre los números binarios A y B, en primer lugar se examina el bit de mayor orden de cada número. Las posibles condiciones son las siguientes:

EJEMPLO 6.6

Determinar las salidas $A = B$, $A > B$ y $A < B$ para los números de entrada mostrados en el comparador de la Figura 6.23.

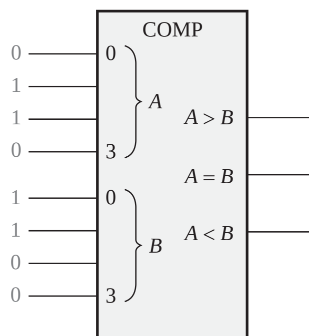


FIGURA 6.23

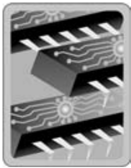
Solución El número que hay en las entradas A es 0110 y el número que hay en las entradas B es 0011. La **salida $A > B$ está a nivel ALTO y las restantes salidas están a nivel BAJO.**

Problema relacionado ¿Cuáles serán las salidas del comparador cuando $A_3A_2A_1A_0 = 1001$ y $B_3B_2B_1B_0 = 1010$?

1. Si $A_3 = 1$ y $B_3 = 0$, entonces A es mayor que B.
2. Si $A_3 = 0$ y $B_3 = 1$, entonces A es menor que B.
3. Si $A_3 = B_3$, entonces tenemos que examinar los siguientes bits de orden inmediatamente inferior.

Estas tres operaciones son válidas para cada posición que ocupen los bits dentro del número. El procedimiento general utilizado en un comparador consiste en comprobar una desigualdad en cualquier posición de bit, comenzando por los bits más significativos (MSB). Cuando se encuentra una desigualdad, la relación entre ambos números queda establecida y cualquier otra desigualdad entre bits con posiciones de orden menor debe ignorarse, ya que podrían indicar una relación entre los números completamente opuesta. *La relación de más alto orden es la que tiene prioridad.*

EL COMPARADOR DE MAGNITUD DE 4 BITS 74HC85



El 74HC85 es un comparador que también se encuentra disponible en otras familias de circuitos integrados. El diagrama de pines y el símbolo lógico se muestran en la Figura 6.24. Observe que este dispositivo tiene todas las entradas y salidas del comparador visto anteriormente y, además, tiene tres entradas en cascada: $A < B$, $A = B$ y $A > B$. Estas entradas permiten utilizar varios comparadores en cascada para la comparación de cualquier número binario con más de cuatro bits. Para ampliar el comparador, las salidas $A < B$, $A = B$ y $A > B$ del comparador de menor orden se conectan en cascada a las entradas del siguiente comparador de orden inmediatamente superior. El comparador de menor orden tiene que tener un nivel ALTO en la entrada $A = B$ y un nivel BAJO en las entradas $A < B$ y $A > B$. Este dispositivo está disponible en otras familias CMOS y TTL. Consulte el sitio web de Texas Instruments en www.ti.com.

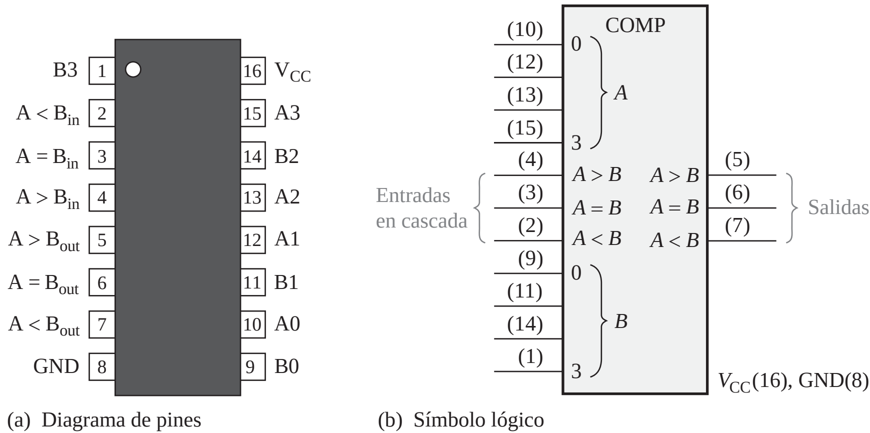


FIGURA 6.24 Diagrama de pines y símbolo lógico del comparador de magnitud de 4 bits 74HC85 (la numeración de los pines se muestra entre paréntesis).

EJEMPLO 6.7

Utilizar comparadores 74HC85 para comparar las magnitudes de dos números de 8 bits. Dibujar los comparadores con sus correspondientes interconexiones.

Solución Se necesitan dos 74HC85 para comparar dos números de 8 bits. Éstos se conectan en cascada como se muestra en la Figura 6.25, empleando una disposición en cascada.

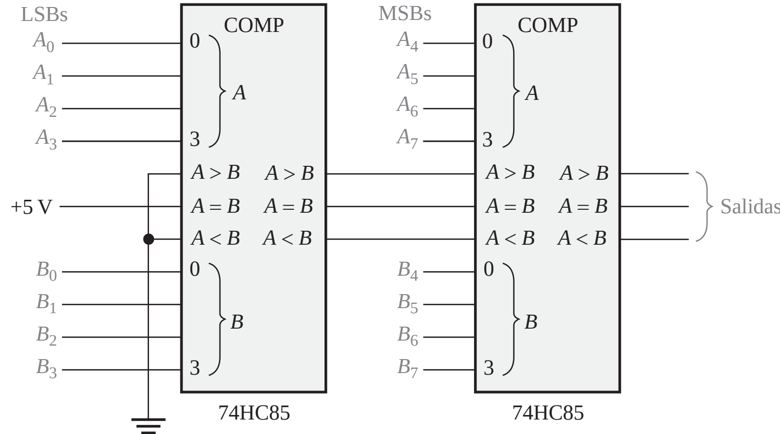


FIGURA 6.25 Comparador de 8 bits formado por dos 74HC85.

Problema relacionado Ampliar el circuito de la Figura 6.25 para realizar un comparador de 16 bits.

CONSEJOS PRÁCTICOS

La mayoría de los dispositivos CMOS contienen circuitería de protección para protegerse frente a daños generados por altas tensiones estáticas o campos eléctricos. Sin embargo, deben tomarse precauciones para evitar la aplicación de cualquier tensión mayor que la tensión máxima nominal. Para un funcionamiento correcto, las tensiones de entrada y salida deberían estar comprendidas entre tierra y V_{CC}. Recuerde también que las entradas no utilizadas deberán conectarse siempre a un nivel lógico apropiado (tierra o V_{CC}). Las salidas no utilizadas pueden dejarse en circuito abierto.

REVISIÓN DE LA SECCIÓN 6.4

1. Los números binarios A = 1011 y B = 1010 se aplican a las entradas de un 74HC85. Determinar las salidas.
2. Los números binarios A = 11001011 y B = 11010100 se aplican al comparador de 8 bits de la Figura 6.25. Determinar el estado de los pines de salida 5, 6 y 7 en cada uno de los 74HC85.

6.5 DECODIFICADORES

La función básica de un decodificador es detectar la presencia de una determinada combinación de bits (código) en sus entradas y señalar la presencia de este código mediante un cierto nivel de salida. En su forma general, un decodificador posee n líneas de entrada para gestionar n bits y en una de las 2^n líneas de salida indica la presencia de una o más combinaciones de n bits. En esta sección, se presentan varios tipos de decodificadores. Los principios básicos se pueden extender a otros decodificadores.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Definir *decodificador*. ■ Diseñar un circuito lógico para decodificar cualquier combinación de bits. ■ Describir el decodificador binario-decimal 74HC154. ■ Ampliar los decodificadores para poder tratar códigos con un número de bits superior. ■ Describir el decodificador BCD a 7-segmentos 74LS47. ■ Explicar la supresión de cero en los displays de 7-segmentos. ■ Utilizar los decodificadores en aplicaciones específicas.



NOTAS INFORMÁTICAS

Una *instrucción* indica a la computadora qué operación debe realizar. Las instrucciones se especifican en código máquina (1s y 0s) y, para que la computadora ejecute una instrucción, ésta debe ser decodificada. La decodificación de las instrucciones es uno de los pasos en la *pipeline* (secuencia de procesamiento) de las instrucciones; los pasos de dicho proceso son: la instrucción se lee desde la memoria (extracción de la instrucción), la instrucción se decodifica, se leen los operandos desde la memoria (extracción de operandos), se ejecuta la instrucción y el resultado se escribe de nuevo en memoria. Básicamente, el procesamiento *pipeline* permite que se comience a procesar la siguiente instrucción antes de haber completado la instrucción actual.

El decodificador binario básico

Supongamos que necesitamos determinar cuándo aparece el número binario 1001 en las entradas de un circuito digital. Se puede utilizar una puerta AND como elemento básico de decodificación, ya que produce una salida a nivel ALTO sólo cuando todas sus entradas están a nivel ALTO. Por tanto, debe asegurarse de que todas las entradas de la puerta AND estén a nivel ALTO cuando se introduce el número 1001, lo cual se puede conseguir invirtiendo los dos bits centrales (cuyos bits son 0), como se muestra en la Figura 6.26.

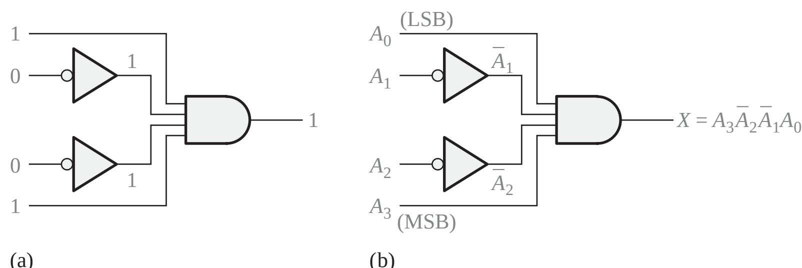


FIGURA 6.26 Lógica de decodificación del código binario 1001 con una salida activa a nivel ALTO.

La ecuación lógica para el **decodificador** de la Figura 6.26(a) se desarrolla como se ilustra en la Figura 6.26(b). Se debe comprobar que la salida es siempre 0 excepto cuando se aplican las entradas $A_0 = 1$, $A_1 = 0$, $A_2 = 0$ y $A_3 = 1$. A_0 es el bit menos significativo y A_3 el más significativo. *En este libro, cuando se representa un número binario o cualquier otro código de pesos, el bit menos significativo siempre es el situado más a la derecha cuando el número se escribe en sentido horizontal, y el de más arriba cuando se escribe en vertical, a menos que se indique lo contrario*.

Si se utiliza una puerta NAND en lugar de una AND, como se muestra en la Figura 6.26, una salida a nivel BAJO indicará la presencia del código binario adecuado, que en este caso es 1001.

EJEMPLO 6.8

Determinar la lógica requerida para decodificar el número binario 1011 de manera que produzca un nivel ALTO en la salida.

Solución La función de decodificación la podemos realizar complementando sólo las variables cuyo valor es 0 en el número binario deseado, tal y como sigue:

$$X = A_3\overline{A}_2A_1A_0 \qquad (1011)$$

Esta función puede implementarse conectando las variables verdaderas (no complementadas) A_0, A_1 y A_3 directamente a las entradas de la puerta AND, e invirtiendo la variable A_2 antes de aplicarla a la puerta AND. La lógica de decodificación se muestra en la Figura 6.27.

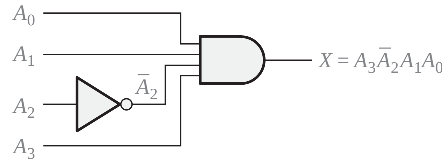


FIGURA 6.27 Lógica de decodificación para generar una salida a nivel ALTO cuando se aplica el código 1011 en las entradas.

Problema relacionado Desarrollar la lógica requerida para detectar el código binario 10010 y generar una salida activa a nivel BAJO.

El decodificador de 4 bits

Para poder decodificar todas las posibles combinaciones de cuatro bits, se necesitan dieciséis puertas de decodificación (2^4=16). Este tipo de decodificador se denomina comúnmente *decodificador de 4 líneas a 16 líneas*, ya que existen cuatro entradas y dieciséis salidas, o también se le llama *decodificador 1 de 16*, ya que para cualquier código dado en las entradas, sólo se activa una de las dieciséis posibles salidas. En la Tabla 6.4 se muestra una lista de los dieciséis códigos binarios y sus correspondientes funciones de decodificación.

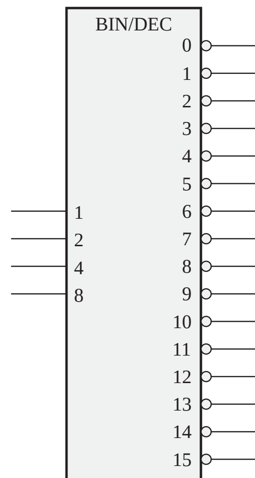


FIGURA 6.28 Símbolo lógico de un decodificador de 4-líneas a 16-líneas (1 de 16).

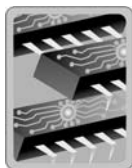
Si se necesita una salida activa a nivel BAJO para cada número decodificado, el decodificador completo se puede implementar mediante puertas NAND e inversores. Para decodificar cada uno de los dieciséis códigos binarios se requieren dieciséis puertas NAND (las puertas AND se pueden usar para producir salidas activas a nivel ALTO).

El símbolo lógico de un decodificador de 4 líneas a 16 líneas (1 de 16) con salidas activas a nivel BAJO se muestra en la Figura 6.28. La etiqueta BIN/DEC indica que una entrada binaria produce su correspondiente salida decimal. Las etiquetas 8, 4, 2 y 1 en las entradas representan los pesos binarios de los bits de entrada ($2^3\ 2^2\ 2^1\ 2^0$).

Dígito decimal	**Entradas binarias**				**Función de decodificación**	**Salidas**															
	A_3	A_2	A_1	A_0		**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	0	0	0	0	$\overline{A}_3\overline{A}_2\overline{A}_1\overline{A}_0$	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	0	0	0	1	$\overline{A}_3\overline{A}_2\overline{A}_1A_0$	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	0	0	1	0	$\overline{A}_3\overline{A}_2A_1\overline{A}_0$	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
3	0	0	1	1	$\overline{A}_3\overline{A}_2A_1A_0$	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1
4	0	1	0	0	$\overline{A}_3A_2\overline{A}_1\overline{A}_0$	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1
5	0	1	0	1	$\overline{A}_3A_2\overline{A}_1A_0$	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1
6	0	1	1	0	$\overline{A}_3A_2A_1\overline{A}_0$	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1
7	0	1	1	1	$\overline{A}_3A_2A_1A_0$	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1
8	1	0	0	0	$A_3\overline{A}_2\overline{A}_1\overline{A}_0$	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1
9	1	0	0	1	$A_3\overline{A}_2\overline{A}_1A_0$	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1
10	1	0	1	0	$A_3\overline{A}_2A_1\overline{A}_0$	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
11	1	0	1	1	$A_3\overline{A}_2A_1A_0$	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1
12	1	1	0	0	$A_3A_2\overline{A}_1\overline{A}_0$	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1
13	1	1	0	1	$A_3A_2\overline{A}_1A_0$	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
14	1	1	1	0	$A_3A_2A_1\overline{A}_0$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
15	1	1	1	1	$A_3A_2A_1A_0$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

TABLA 6.4 Funciones de decodificación y tabla de verdad para un decodificador de 4-líneas a 16-líneas con salidas activas a nivel BAJO.

EL DECODIFICADOR 1 DE 16 74HC154



El 74HC154 es un buen ejemplo de un decodificador en circuito integrado. Su símbolo lógico se muestra en la Figura 6.29. En este tipo de dispositivo existe una función de activación (*enable*, *EN*), que se implementa mediante una puerta NOR utilizada como negativa-AND. En las entradas de selección del chip, $\overline{CS}_1$ y $\overline{CS}_2$, se requiere un nivel BAJO para obtener en la salida de la puerta de activación (EN, *enable*) un nivel ALTO. La salida de la puerta de activación se conecta a una entrada de *cada* puerta NAND del decodificador, por lo que debe estar a nivel ALTO para que las puertas NAND se activen. Si la puerta de activación no se activa mediante un nivel BAJO en ambas entradas, entonces las dieciséis salidas (Y) del decodificador estarán a nivel ALTO independientemente del estado de las cuatro variables de entrada A_0, A_1, A_2 y A_3. Este dispositivo puede estar disponible en otras familias CMOS o TTL. Consulte el sitio web de Texas Instruments en www.ti.com.

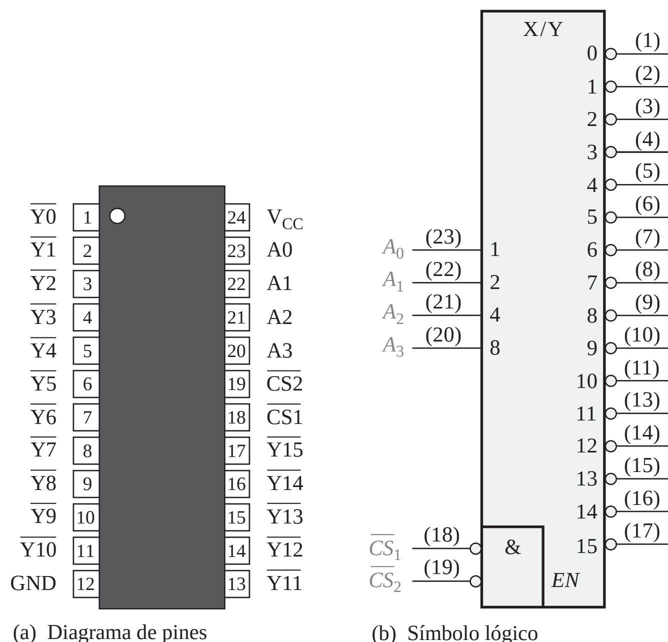


FIGURA 6.29 Diagrama de pines y símbolo lógico para el decodificador 1 de 16 74HC154.

EJEMPLO 6.9

Una cierta aplicación requiere decodificar un número de 5 bits. Utilizar decodificadores 74HC154 para implementar el circuito lógico. El número binario se representa de la forma:

$$A_4A_3A_2A_1A_0$$

Solución Dado que el 74HC154 puede procesar únicamente cuatro bits, habrá que usar dos decodificadores para los cinco bits. El quinto bit, A_4, está conectado a las entradas de selección del chip, $\overline{CS}_1$ y $\overline{CS}_2$, de uno de los decodificadores y $\overline{A}_4$ se conecta a las entradas de activación $\overline{CS}_1$ y $\overline{CS}_2$ del otro decodificador, como se muestra en la Figura 6.30.

Cuando el número decimal es 15 o menor, $A_4 = 0$, el decodificador de menor orden se activa y el de mayor orden se desactiva.

Cuando el número decimal es mayor que 15, $A_4 = 1$, luego $\overline{A}_4 = 0$, lo que hace que se active el decodificador de orden superior y se desactive el de orden inferior.

Problema relacionado En la Figura 6.30, determinar la salida que se activa al introducir el código binario de entrada:

10110

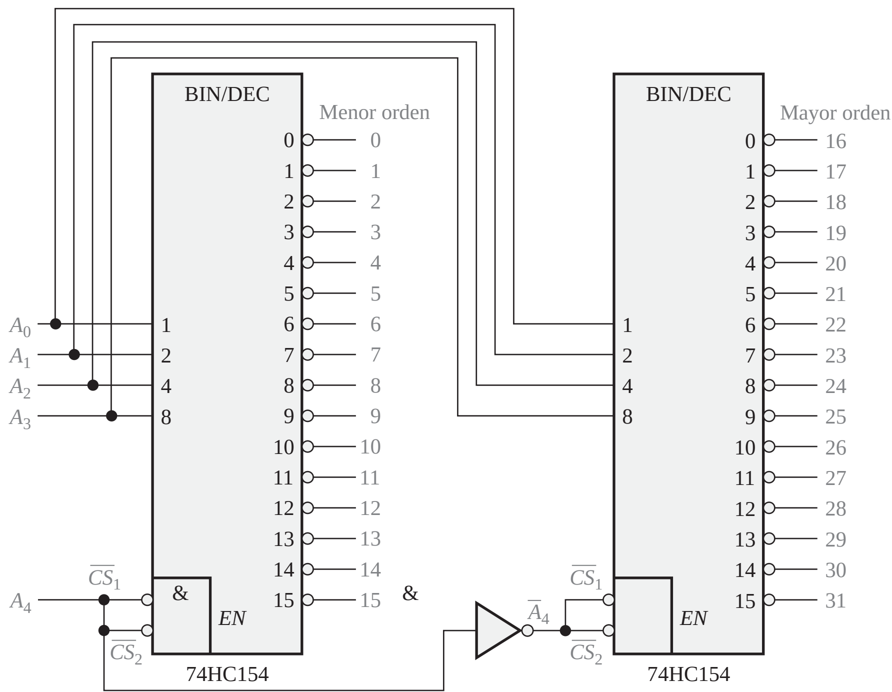


FIGURA 6.30 Decodificador de 5 bits construido con dos 74HC154.

Aplicación

Los decodificadores se utilizan en muchos tipos de aplicaciones. Un ejemplo es la selección de entradas y salidas en las computadoras, como se muestra en el diagrama general de la Figura 6.31.

Las computadoras se tienen que comunicar con una gran variedad de dispositivos externos, denominados *periféricos*, enviando y/o recibiendo datos a través de lo que se conoce como puertos de entrada/salida (E/S). Estos dispositivos externos incluyen impresoras, modems, escáneres, unidades de disco externas, teclados, monitores y otras computadoras. Como se indica en la Figura 6.31, se emplea un decodificador para seleccionar el puerto de E/S determinado por la computadora, de forma que los datos puedan ser enviados o recibidos desde algún dispositivo externo concreto.

Cada puerto de E/S tiene un número, denominado dirección, que lo identifica unívocamente. Cuando la computadora desea comunicarse con algún dispositivo en particular, envía el código de dirección apropiado del puerto de E/S al que está conectado el dispositivo en cuestión. Esta dirección binaria del puerto se decodifica, activándose la salida del decodificador apropiada que habilita el correspondiente puerto de E/S.

Como se muestra en la Figura 6.31, los datos binarios se transfieren dentro de la computadora a través de un bus de datos, que consiste en un conjunto de líneas paralelas. Por ejemplo, un bus de 8 bits consta de ocho líneas paralelas que pueden transmitir un byte de datos de una sola vez. El bus de datos está conectado a todos los puertos de E/S, pero los datos que son recibidos o transmitidos sólo pasarán a través del puerto que se encuentre activado por el decodificador de direcciones de puertos.

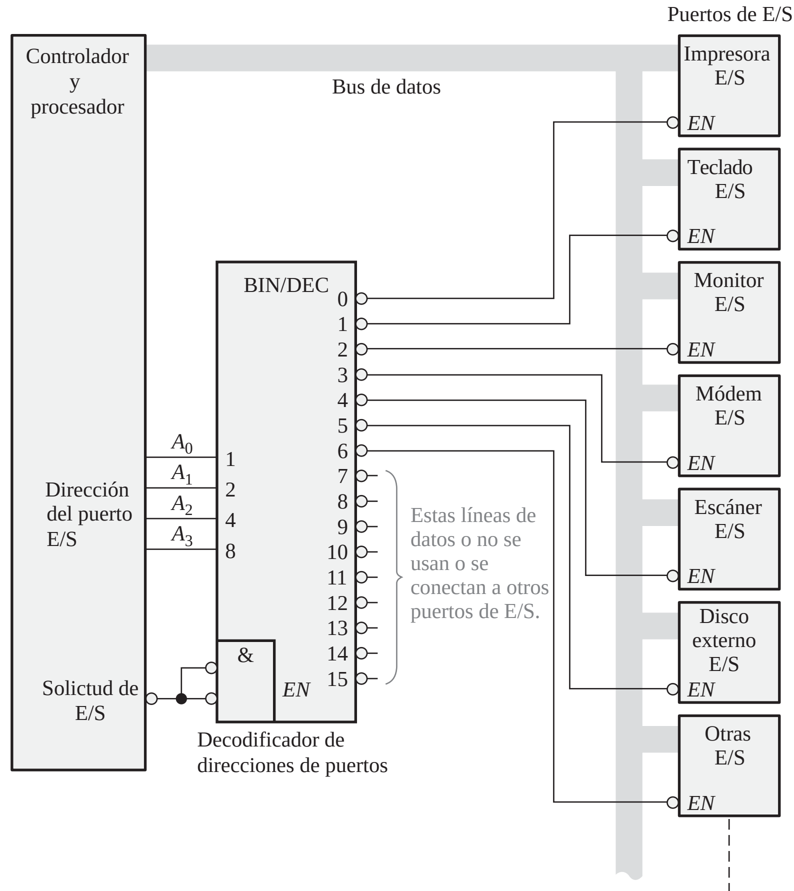


FIGURA 6.31 Un sistema simplificado de puertos de E/S con un decodificador de direcciones de puerto con sólo cuatro líneas de dirección.

El decodificador BCD a decimal

Un decodificador BCD a decimal convierte cada código BCD (código 8421) en uno de los diez posibles dígitos decimales. Frecuentemente, se le denomina *decodificador de 4-líneas a 10-líneas* o *decodificador 1 de 10*.

El método de implementación es el mismo que hemos visto anteriormente para el decodificador de 4-líneas a 16-líneas, excepto que ahora sólo se requieren diez puertas decodificadoras, dado que el código BCD sólo representa los diez dígitos decimales de 0 a 9. En la Tabla 6.5 se muestra una lista de los diez códigos BCD y sus correspondiente funciones de decodificación. Cada una de estas funciones se implementa mediante puertas NAND para proporcionar salidas activas a nivel BAJO. Si se requirieran salidas activas a nivel ALTO, se utilizarían puertas AND para la decodificación. La lógica es idéntica a la de las diez primeras puertas del decodificador de 4-líneas a 16-líneas (*véase* la Tabla 6.4).

EJEMPLO 6.10

El 74HC42 es un CI decodificador BCD-decimal. Su símbolo lógico se muestra en la Figura 6.32. Dibujar las señales de salida si se aplican las señales de entrada de la Figura 6.33(a) a las entradas del 74HC42.

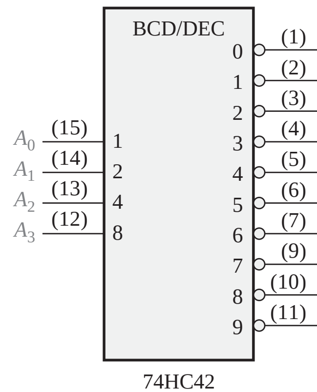


FIGURA 6.32 El decodificador BCD a decimal 74HC42.

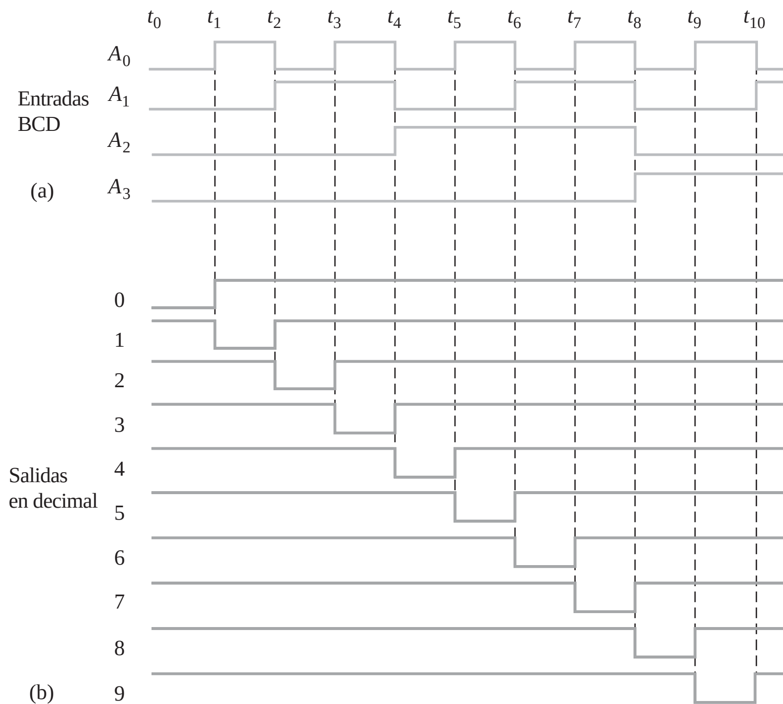


FIGURA 6.33

Solución Las formas de onda de salida se muestran en la Figura 6.33(b). Como puede ver, las entradas al decodificador constituyen una secuencia de dígitos de 0 a 9. Las formas de onda de salida del diagrama de tiempos indican dicha secuencia de valores decimales.

Problema relacionado Construir un diagrama de tiempos que muestre las señales de entrada y de salida para el caso en que la secuencia binaria de entrada origine los siguientes números decimales: 0, 2, 4, 6, 8, 1, 3, 5 y 9.

Dígito decimal	Código BCD A_3	A_2	A_1	A_0	Función de decodificación
0	0	0	0	0	$\overline{A}_3\overline{A}_2\overline{A}_1\overline{A}_0$
1	0	0	0	1	$\overline{A}_3\overline{A}_2\overline{A}_1A_0$
2	0	0	1	0	$\overline{A}_3\overline{A}_2A_1\overline{A}_0$
3	0	0	1	1	$\overline{A}_3\overline{A}_2A_1A_0$
4	0	1	0	0	$\overline{A}_3A_2\overline{A}_1\overline{A}_0$
5	0	1	0	1	$\overline{A}_3A_2\overline{A}_1A_0$
6	0	1	1	0	$\overline{A}_3A_2A_1\overline{A}_0$
7	0	1	1	1	$\overline{A}_3A_2A_1A_0$
8	1	0	0	0	$A_3\overline{A}_2\overline{A}_1\overline{A}_0$
9	1	0	0	1	$A_3\overline{A}_2\overline{A}_1A_0$

TABLA 6.5 Funciones de decodificación BCD.

El decodificador BCD a 7-segmentos

El decodificador BCD a 7-segmentos acepta el código BCD en sus entradas y proporciona salidas capaces de excitar un display de 7-segmentos para generar un dígito decimal. En la Figura 6.34 se muestra el diagrama lógico de un decodificador básico de 7-segmentos.

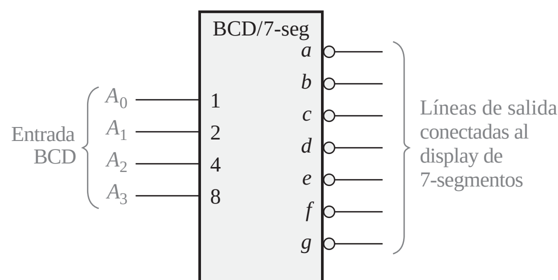
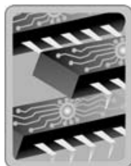


FIGURA 6.34 Símbolo lógico de un decodificador/controlador BCD a 7-segmentos con salidas activas a nivel BAJO.

EL DECODIFICADOR /CONTROLADOR BCD A 7-SEGMENTOS 74LS47



El 74LS47 es un ejemplo de circuito integrado que decodifica una entrada BCD y controla un display de 7-segmentos. Además de estas características de decodificación y control, el 74LS47 posee características adicionales, como las indicadas en el símbolo lógico de la Figura 6.35 por las funciones $\overline{LT}, \overline{RBI}, \overline{BI}/\overline{RBO}$. Como indican los círculos del símbolo lógico, todas las salidas (de *a* a *g*) son activas a nivel BAJO, al igual que lo son $\overline{LT}$ (*Lamp Test*, entrada de comprobación), *RB*I (*Ripple Blanking Input*) y $\overline{BI}/\overline{RBO}$. (*Blanking Input/Ripple Blanking Output*). Las salidas pueden controlar directamente un display de 7-segmentos en ánodo común. Recuerde que los displays de 7-segmentos se trataron en el Capítulo 4. Además de decodificar una entrada BCD y generar las apropiadas salidas para 7-segmentos, el 74LS47 posee las funciones de entra-

da de comprobación y de supresión de cero. Este dispositivo puede estar disponible en otras familias CMOS o TTL. Consulte el sitio web de Texas Instruments en www.ti.com.

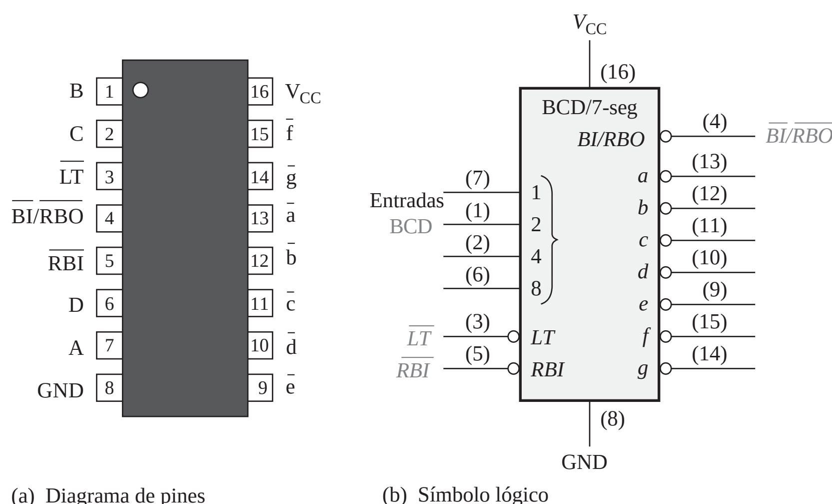


FIGURA 6.35 Diagrama de pines y símbolo lógico para el codificador/controlador BCD a 7-segmentos 74LS47.

Entrada de comprobación. Cuando se aplica un nivel BAJO a la entrada $\overline{LT}$ y la entrada $\overline{BI}/\overline{RBO}$.está a nivel ALTO, se encienden los 7 segmentos del display. La entrada de comprobación se utiliza para verificar que ninguno de los segmentos está fundido.

▲ *La supresión de cero da como resultado que no se muestren en un display los ceros anteriores o posteriores de un número.*

Supresión de cero. La **supresión de cero** es una característica utilizada en displays de varios dígitos para eliminar los ceros innecesarios. Por ejemplo, en un display de 6 dígitos, el número 6.4 podría mostrarse como 006.400 si no se eliminaran los ceros. La supresión de ceros al principio de un número recibe el nombre de *supresión anterior de cero*, mientras que si son los últimos los que se eliminan se denomina *supresión posterior de cero*. Tenga en cuenta que únicamente se eliminan los ceros que no son esenciales. Gracias a la supresión de cero, el número 030.080 se visualiza como 30.08 (los ceros esenciales permanecen).

La supresión de cero en el 74LS47 se logra utilizando las funciones $\overline{RBI}$ y $\overline{BI}/\overline{RBO}$. $\overline{RBI}$ es la entrada de borrado en cascada y $\overline{RBO}$ es la salida de borrado en cascada del 74LS47, las cuales se utilizan para la supresión de cero. $\overline{BI}$ es la entrada de borrado y comparte el mismo pin con $\overline{RBO}$. En otras palabras, el pin $\overline{BI}/\overline{RBO}$ puede utilizarse como salida o como entrada. Cuando se emplea como $\overline{BI}$ (entrada de borrado), todas las salidas están a nivel ALTO (segmentos desactivados) cuando $\overline{BI}$ está a nivel BAJO, anulando el resto de entradas. La función $\overline{BI}$ no forma parte de las características de supresión de cero del dispositivo.

Todas las salidas del decodificador correspondientes a los segmentos se encuentran inactivas (nivel ALTO) cuando se introduce el código cero (0000) en sus entradas BCD siempre que su $\overline{RBI}$ está a nivel BAJO. Esto origina que el display no muestre nada y que la salida $\overline{RBO}$ esté a nivel BAJO.

El diagrama lógico de la Figura 6.36(a) ilustra la supresión anterior de cero para un número entero. El dígito más significativo (el de más a la izquierda) desaparece siempre que se introduzca el código del 0 en sus entradas BCD, ya que la $\overline{RBI}$ del decodificador

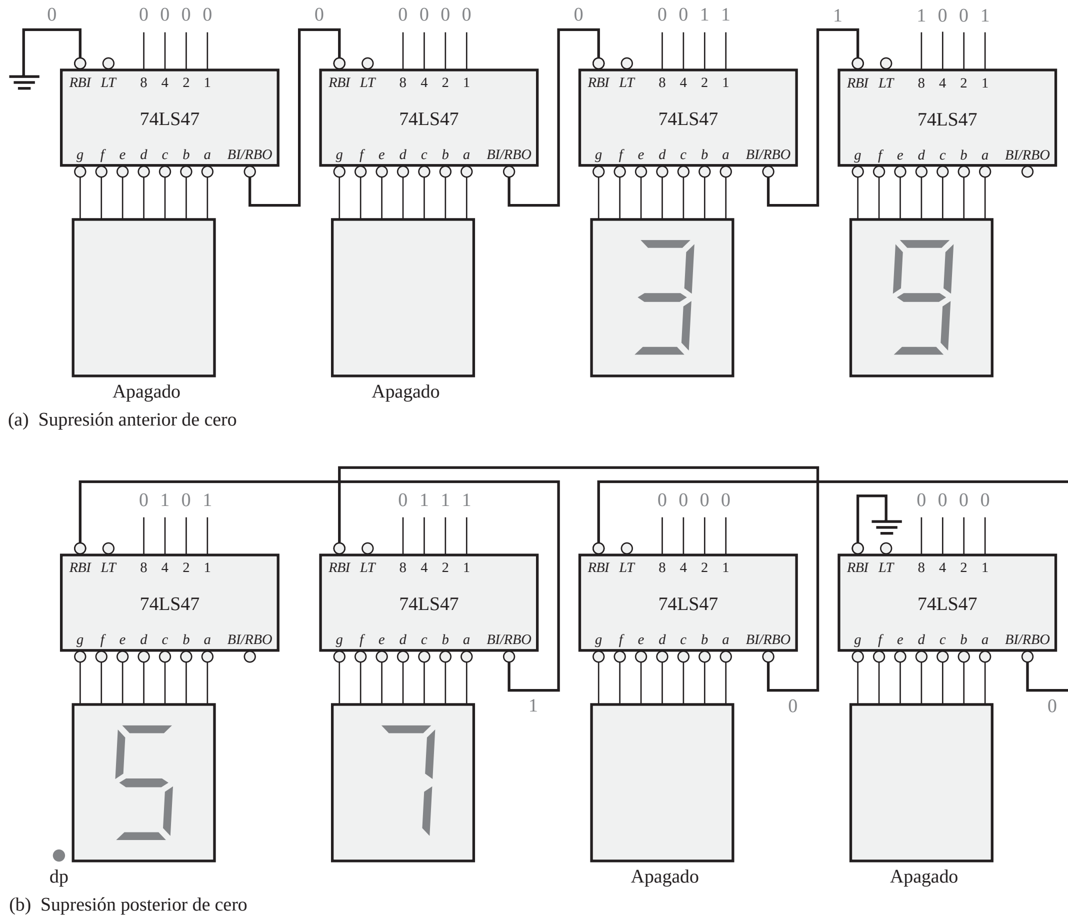


FIGURA 6.36 Ejemplos de supresión de cero mediante un decodificador/controlador BCD a 7-segmentos 74LS47.

de orden más alto se conecta a tierra de forma que siempre esté a nivel BAJO. La salida $\overline{RBO}$ de cada decodificador se conecta a la entrada $\overline{RBI}$ del siguiente decodificador de menor orden, de manera que se eliminan todos los ceros a la izquierda del primer dígito que no sea cero. Por ejemplo, en la parte (a) de la figura, los dos dígitos más significativos son cero y, por tanto, se eliminan. Los dos dígitos restantes, 3 y 9, se visualizan en el display.

El diagrama lógico de la Figura 6.36(b) ilustra la supresión posterior de cero para un número fraccionario. El dígito de menor orden (el de más a la derecha) se suprime siempre que se introduzca un código nulo en sus entradas BCD, ya que la entrada $\overline{RBI}$ está conectada a masa. La salida $\overline{RBO}$ de cada decodificador está conectada a la entrada $\overline{RBI}$ del decodificador siguiente de orden superior, de forma que todos los ceros a la derecha del primer dígito no nulo son eliminados. En la parte (b) de la figura, los dos dígitos menos significativos son cero, luego se eliminan. Los dos dígitos restantes, 5 y 7, se muestran en el display. Para combinar la supresión anterior de cero con la posterior en un mismo display y, además, tener posibilidad de introducir puntos decimales, necesitamos circuitos lógicos adicionales.

REVISIÓN DE LA SECCIÓN 6.5

1. Un decodificador de 3-líneas a 8-líneas se puede utilizar como decodificador octal a decimal. Cuando se introduce el número binario 101 en sus entradas, ¿qué línea de salida se activa?
2. ¿Cuántos decodificadores 1 de 16 74HC154 son necesarios para decodificar un número binario de 6 bits?
3. ¿Qué elegiríamos para controlar un display de 7-segmentos con cátodo común, un decodificador/controlador con salidas activas a nivel BAJO o con salidas activas a nivel ALTO?

6.6 CODIFICADORES

Un ***codificador*** es un circuito lógico combinacional que, esencialmente, realiza la función "inversa" del decodificador. Un codificador permite que se introduzca en una de sus entradas un nivel activo que representa un dígito, como puede ser un dígito decimal u octal, y lo convierte en una salida codificada, como BCD o binario. Los codificadores se pueden diseñar también para codificar símbolos diversos y caracteres alfabéticos. El proceso de conversión de símbolos comunes o números a un formato codificado recibe el nombre de *codificación*.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Determinar la lógica de un codificador decimal. ■ Explicar la finalidad de la característica de prioridad en los codificadores. ■ Describir el codificador de prioridad decimal a BCD 74HC147. ■ Describir el codificador de prioridad octal a binario 74LS148. ■ Ampliar un codificador. ■ Utilizar los codificadores en aplicaciones específicas.

Codificador decimal-BCD

Este tipo de codificador tiene diez entradas, una para cada dígito decimal, y cuatro salidas que corresponden al código BCD, como se muestra en la Figura 6.37. Este es un codificador básico de 10-líneas a 4-líneas.

El código BCD (8421) se muestra en la Tabla 6.6. A partir de esta tabla podemos determinar la relación entre cada bit BCD y los dígitos decimales, con el fin de analizar la lógica. Por ejemplo, el bit más significativo del código BCD, A_3, es siempre un 1 para los dígitos decimales 8 o 9. La expresión OR para el bit A_3 en función de los dígitos decimales puede por tanto escribirse como:

$$A_3 = 8 + 9$$

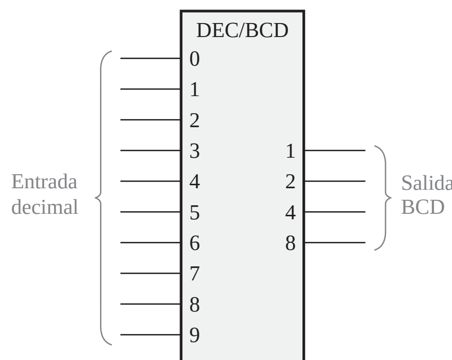


FIGURA 6.37 Símbolo lógico de un codificador decimal a BCD.

Dígito decimal	Código BCD			
	A_3	A_2	A_1	A_0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

TABLA 6.6

El bit A_2 es siempre un 1 para los dígitos decimales 4, 5, 6 o 7 y puede expresarse como una función OR de la manera siguiente:

$$A_2 = 4 + 5 + 6 + 7$$

El bit A_1 es siempre un 1 para los dígitos decimales 2, 3, 6 o 7 y puede expresarse como:

$$A_1 = 2 + 3 + 6 + 7$$

Finalmente, A_0 es siempre un 1 para los dígitos 1, 3, 5, 7 o 9. La expresión para A_0 es:

$$A_0 = 1 + 3 + 5 + 7 + 9$$

Ahora vamos a implementar el circuito lógico necesario para codificar en código BCD cada dígito decimal, utilizando las expresiones lógicas que se acaban de desarrollar. Consiste simplemente en aplicar la operación OR a los dígitos decimales de entrada apropiados, para así formar cada salida BCD. La lógica del codificador que resulta de estas expresiones se muestra en la Figura 6.38.

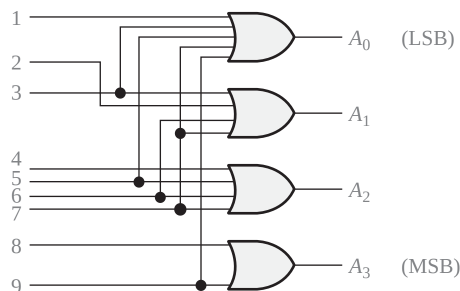
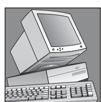


FIGURA 6.38 Diagrama lógico básico de un codificador decimal-BCD. No se necesita una entrada para el dígito 0, ya que las salidas BCD están todas a nivel BAJO cuando no hay entradas a nivel ALTO.



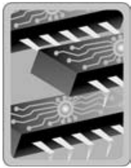
NOTAS INFORMÁTICAS

Se puede pensar en un *ensamblador* como en un codificador de software, ya que interpreta las instrucciones mnemónicas con las que se ha escrito un programa y lleva a cabo la codificación necesaria para convertir cada mnemónico en una instrucción de código máquina (series de 1s y 0s), que la computadora puede entender. Ejemplos de mnemónicos de instrucciones son ADD, MOV (*move data*, desplazamiento de datos), MUL (multiplicación), XOR, JMP (*jump*, salto) y OUT (salida a un puerto).

El funcionamiento básico del circuito de la Figura 6.38 es el siguiente: cuando aparece un nivel ALTO en *una* de las líneas de entrada correspondientes a los dígitos decimales, se generan los niveles apropiados en las cuatro líneas BCD de salida. Por ejemplo, si la línea de entrada 9 está a nivel ALTO (suponiendo que todas las demás entradas están a nivel BAJO), esta condición producirá un nivel ALTO en las salidas A_0 y A_3, y un nivel BAJO en A_1 y A_2, que es el código BCD (1001) del número decimal 9.

Codificador con prioridad decimal a BCD. Este tipo de codificador realiza la misma función de codificación básica que hemos visto anteriormente. Además, un ***codificador con prioridad*** ofrece una flexibilidad adicional en lo relativo a que puede utilizarse en aplicaciones que requieren detección de prioridad. La función de prioridad significa que el codificador producirá una salida BCD correspondiente al *dígito decimal de entrada de más alto orden* que se encuentre activo, e ignorará cualquier otra entrada de menor orden que esté activa. Por ejemplo, si las entradas 6 y 3 se encuentran activas, la salida BCD será 0110 (que representa al número decimal 6).

EL CODIFICADOR DECIMAL-BCD 74HC147



El 74HC147 es un codificador con prioridad con entradas activas a nivel BAJO (0) para los dígitos decimales del 1 al 9, y salidas BCD activas a nivel BAJO, como se indica en el símbolo lógico de la Figura 6.39. Una salida BCD cero se consigue cuando ninguna de las entradas está activa. La numeración de los pines del dispositivo se muestra entre paréntesis. Este dispositivo puede estar disponible en otras familias CMOS o TTL. Consulte el sitio web de Texas Instruments en www.ti.com.

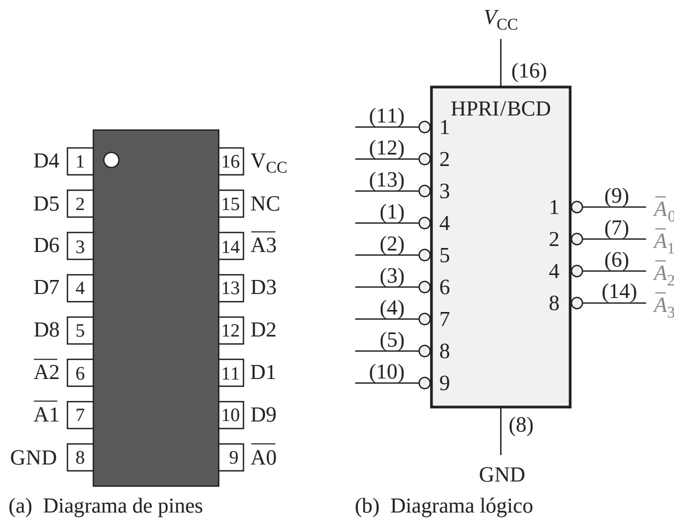


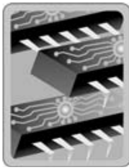
FIGURA 6.39 Diagrama de pines y símbolo lógico del codificador con prioridad decimal-BCD 74HC147 (HPRI, *highest value input has priority*, la entrada de mayor valor tiene prioridad).

EJEMPLO 6.11

Si tenemos niveles BAJOS en los pines 1, 4 y 13 del 74HC147 que se muestra en la Figura 6.39, indicar el estado de sus cuatro salidas. Todas las demás entradas están a nivel ALTO.

Solución	El pin 4 es el dígito decimal de orden más alto que tiene una entrada a nivel BAJO y representa el número decimal 7. Por tanto, los niveles de salida indican el código BCD para el decimal 7, donde $\overline{A}_0$ es el bit menos significativo (LSB) y $\overline{A}_3$ es el bit más significativo (MSB). La salida $\overline{A}_0$ es un nivel BAJO, $\overline{A}_1$ es BAJO, $\overline{A}_2$ es un nivel BAJO y $\overline{A}_3$ es un nivel ALTO.
Problema relacionado	¿Cuáles son las salidas del 74HC147 si todas sus entradas están a nivel BAJO? ¿Y si todas están a nivel ALTO?

EL CODIFICADOR 8-LÍNEAS A 3-LÍNEAS 74LS148



El 74LS148 es un codificador con prioridad que tiene ocho entradas activas a nivel BAJO y tres salidas binarias activas a nivel BAJO, como se muestra en la Figura 6.40. Este dispositivo se puede utilizar para convertir entradas octales (recuerde que los dígitos octales son del 0 al 7) en código binario de 3 bits. Para activar este dispositivo, la entrada de activación, (*Enable Input*, *EI*) tiene que estar activa a nivel BAJO. También tiene una *EO* (salida de activación, *Enable Output*) y una salida *GS* para permitir la ampliación. La salida *EO* está a nivel BAJO cuando la entrada *EI* está a nivel BAJO y ninguna de las entradas (de 0 a 7) se encuentra activada. *GS* está a nivel BAJO cuando *EI* está a nivel BAJO y cualquiera de las entradas se encuentra activada. Este dispositivo puede estar disponible en otras familias CMOS o TTL. Consulte el sitio web de Texas Instruments en www.ti.com.

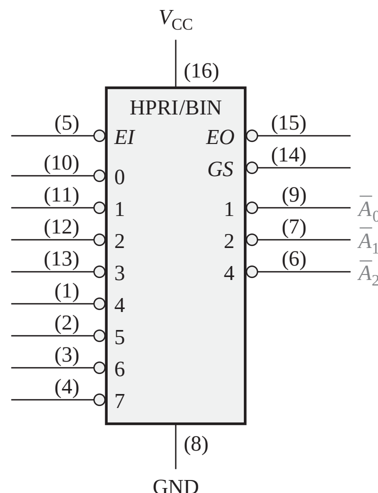


FIGURA 6.40 Símbolo lógico del codificador de 8 líneas a 3 líneas 74LS148.

El 74LS148 puede ser ampliado a un codificador de 16-líneas a 4-líneas conectando la salida *EO* del codificador de mayor orden a la entrada *EI* del codificador de menor orden, y aplicando la operación negativa-OR a las correspondientes salidas binarias, como se muestra en la Figura 6.41. La salida *EO* se utiliza como cuarto y más significa-

tivo bit. Esta configuración particular produce salidas activas a nivel ALTO para los números binarios de cuatro bits.

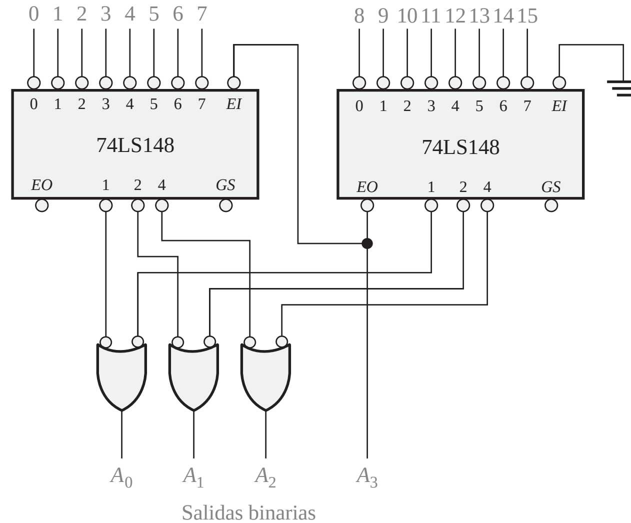


FIGURA 6.41 Un decodificador de 16 líneas a 4 líneas usando dispositivos 74LS148 y lógica externa.

Aplicación

El típico ejemplo de aplicación es un codificador de teclado. Por ejemplo, los diez dígitos decimales del teclado de una computadora tienen que codificarse para poder ser procesados por el circuito lógico. Cuando se pulsa una de las teclas, el dígito decimal se codifica a su correspondiente código BCD. La Figura 6.42 muestra la disposición de un sencillo codificador de teclado que utiliza un codificador con prioridad 74HC147. Las teclas se representan mediante diez pulsadores, conectados cada uno de ellos a una **resistencia de *pull-up*** (resistencia de conexión a la alimentación +V). Las resistencias de *pull-up* aseguran que la línea esté a nivel ALTO cuando no haya ninguna tecla pulsada. Cuando se pulsa una tecla, la línea se conecta a tierra y se aplica un nivel BAJO a la correspondiente entrada del codificador. La tecla cero no está conectada, ya que la salida BCD es cero cuando ninguna de las otras teclas está pulsada.

La salida complementada BCD del codificador se conecta a un dispositivo de almacenamiento, de forma que los sucesivos códigos BCD se almacenan hasta que se haya introducido el número completo. En los siguientes capítulos veremos los métodos de almacenamiento de números BCD y de datos binarios.

REVISIÓN DE LA SECCIÓN 6.6

1. Supongamos que, a las entradas 2 y 9 del circuito de la Figura 6.38 se les aplica un nivel ALTO.
   (a) ¿Cuáles son los estados de las líneas de salida?
   (b) ¿Representa esto un código BCD válido?
   (c) ¿Cuál es la restricción de la lógica del codificador de la Figura 6.38?
2. **(a)** ¿Cuál es la salida $\overline{A}_3\overline{A}_2\overline{A}_1\overline{A}_0$ cuando se aplican niveles BAJOS de tensión a los pines 1 y 5 del 74HC147 de la Figura 6.39?
   (b) ¿Qué representa esta salida?

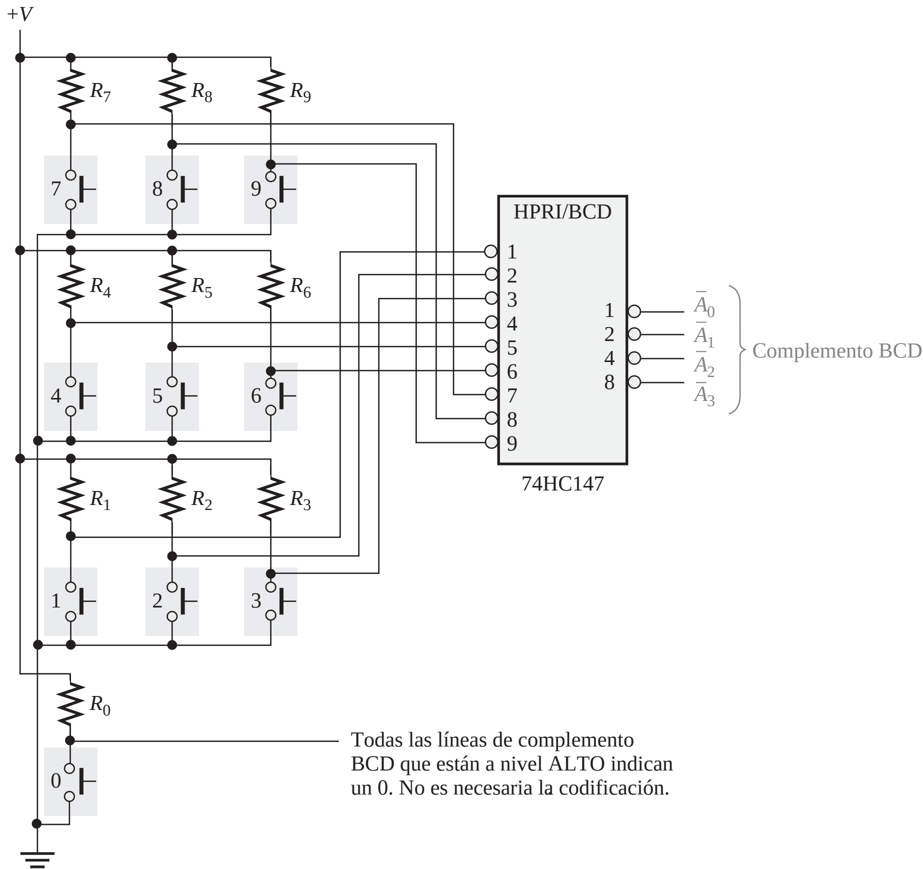


FIGURA 6.42 Codificador de un teclado simplificado.

6.7 CONVERTIDORES DE CÓDIGO

En esta sección, vamos a examinar algunos métodos que utilizan circuitos lógicos combinacionales para pasar de un código a otro.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Explicar el proceso de conversión de BCD a binario. ■ Utilizar puertas OR-exclusiva en la conversión entre código binario y código Gray.

Conversión BCD-binario

Uno de los métodos de conversión de código BCD a binario utiliza circuitos sumadores. El proceso básico de conversión consiste en lo siguiente:

1. El valor, o peso, de cada uno de los bits de un número BCD se representa por un número binario.
2. Se suman todas las representaciones binarias de los pesos de los bits del número BCD que son 1.
3. El resultado de la suma es el equivalente binario del número BCD.

Una manera más concisa de expresar esta operación es:

Para obtener el número binario completo hay que sumar los números binarios que representan los pesos de los bits del número BCD.

Examinemos un código BCD de 8 bits (uno que representa un número decimal de 2 dígitos) para comprender la relación entre el código binario y el BCD. Por ejemplo, ya sabemos que el número decimal 87 puede expresarse en BCD como sigue:

$$\underbrace{1000}_{8}\ \underbrace{0111}_{7}$$

El grupo de 4 bits más a la izquierda representa 80 y el grupo de 4 bits más a la derecha representa 7. Es decir, el grupo más a la izquierda tiene un peso de 10 y el grupo más a la derecha tiene un peso de 1. Dentro de cada grupo, el peso binario de cada bit es el siguiente:

	Dígito de las decenas				**Dígito de las unidades**			
Peso	80	40	20	10	8	4	2	1
Designación de bit:	B_3	B_2	B_1	B_0	A_3	A_2	A_1	A_0

El equivalente binario de cada bit BCD es un número binario que representa el peso de cada bit dentro del número BCD completo. Esta representación se muestra en la Tabla 6.7.

Bit BCD	**Peso BCD**	**(MSB) 64**	**32**	**Representación binaria 16**	**8**	**4**	**2**	**(LSB) 1**
A_0	1	0	0	0	0	0	0	1
A_1	2	0	0	0	0	0	1	0
A_2	4	0	0	0	0	1	0	0
A_3	8	0	0	0	1	0	0	0
B_0	10	0	0	0	1	0	1	0
B_1	20	0	0	1	0	1	0	0
B_2	40	0	1	0	1	0	0	0
B_3	80	1	0	1	0	0	0	0

TABLA 6.7 Representación binaria de los pesos de los bits BCD.

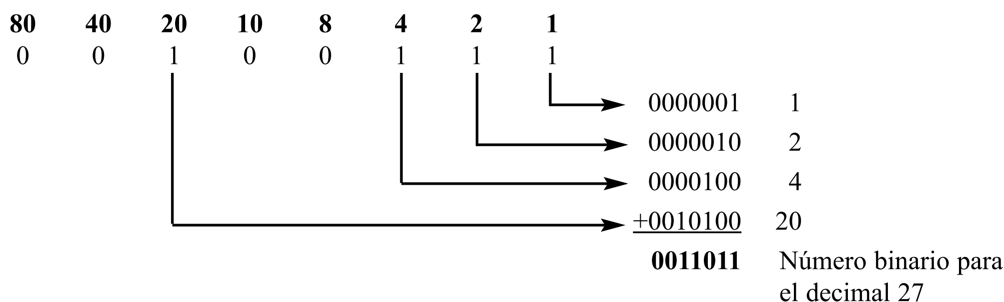
Si las representaciones binarias de los pesos de todos los 1s del número BCD se suman, el resultado es el número binario que corresponde al número BCD. El Ejemplo 6.12 de la página siguiente ilustra esto.

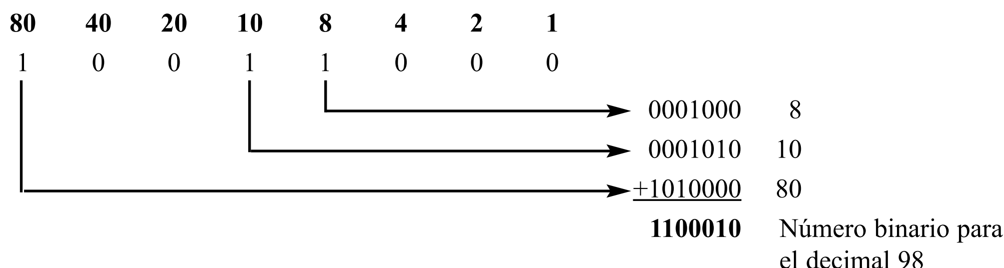
Teniendo este procedimiento básico en mente, vamos a determinar cómo podemos implementar este proceso mediante circuitos lógicos. Una vez que se determina la representación binaria de cada 1 del número BCD, se pueden emplear circuitos sumadores para sumar los 1s de cada columna de la representación binaria. Los 1s aparecen en una determinada columna, sólo cuando los correspondientes bits BCD valen 1. Por tanto, la aparición de un 1 BCD puede utilizarse para generar el 1 binario apropiado en la columna correspondiente de la estructura del sumador. Para manejar un número decimal de dos cifras (dos potencias de diez), en código BCD, necesitamos ocho líneas BCD de entrada y siete líneas binarias de salida. Se necesitan siete bits binarios para representar números de 0 a 99.

EJEMPLO 6.12

Convertir a binario los números BCD 00100111 (27 decimal) y 10011000 (98 decimal).

Solución Escribir la representación binaria de los pesos de todos los 1s que aparecen en los números y, a continuación, sumarlos todos.





Problema relacionado Explicar el proceso de conversión del número BCD 01000001 a binario.

Conversión binario-Gray y Gray-binario

El proceso básico de conversión de código Gray a binario se ha tratado en el Capítulo 2. Ahora vamos a ver cómo se pueden utilizar puertas OR-exclusiva en estas conversiones. Los dispositivos lógicos programables (PLD) también se pueden utilizar para realizar estas conversiones de código. La Figura 6.43 muestra un convertidor de 4 bits binarios a código Gray, y la Figura 6.44 ilustra un convertidor de 4 bits Gray a binario.

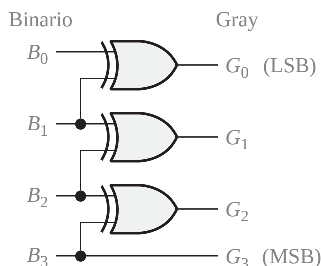


FIGURA 6.43 Lógica de conversión de 4 bits binarios a Gray.

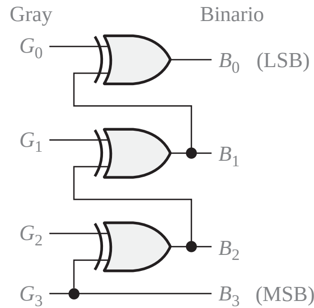


FIGURA 6.44 Lógica de conversión de 4 bits Gray a binario.

EJEMPLO 6.13

(a) Convertir el número binario 0101 a código Gray utilizando puertas OR-exclusiva.
(b) Convertir el código Gray 1011 a binario utilizando puertas OR-exclusiva.

Solución **(a)** 0101_2 es 0111 en código Gray. *Véase* la Figura 6.45(a).
(b) 1011 en código Gray es 1101_2. *Véase* la Figura 6.45(b).

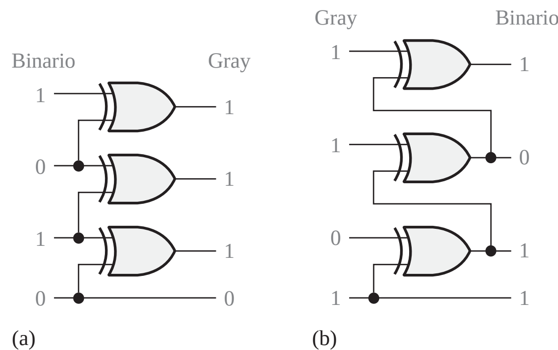


FIGURA 6.45

Problema relacionado ¿Cuántas puertas OR-exclusiva son necesarias para convertir 8 bits binarios a código Gray?

REVISIÓN DE LA SECCIÓN 6.7

1. Convertir el número BCD 10000101 a binario.
2. Dibujar el diagrama lógico para convertir a código Gray un número binario de 8 bits.

6.8 MULTIPLEXORES (SELECTORES DE DATOS)

Un ***multiplexor*** (MUX) es un dispositivo que permite dirigir la información digital procedente de diversas fuentes a una única línea para ser transmitida a través de dicha línea a un destino común. El

multiplexor básico posee varias líneas de entrada de datos y una única línea de salida. También posee entradas de selección de datos, que permiten conmutar los datos digitales provenientes de cualquier entrada hacia la línea de salida. A los multiplexores también se les conoce como selectores de datos.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Explicar el funcionamiento básico de un multiplexor. ■ Describir los multiplexores 74LS151 y 74HC157. ■ Ampliar un multiplexor para poder manejar mayor cantidad de entradas de datos. ■ Utilizar un multiplexor como generador de funciones lógicas.

▲ *En un multiplexor, los datos procedentes de varias líneas pasan a una sola línea.*

El símbolo lógico de un multiplexor (MUX) de cuatro entradas se muestra en la Figura 6.46. Observe que dispone de dos líneas de selección de datos, dado que con dos bits se puede seleccionar cualquiera de las cuatro líneas de entrada de datos.

En la Figura 6.46, un código binario de dos bits en las entradas de selección de datos (S) va a permitir que los datos de la entrada seleccionada pasen a la salida de datos. Si aplicamos un 0 binario ($S_1 = 0$ y $S_0 = 0$) a las líneas de selección de datos, los datos de la entrada D_0 aparecerán en la línea de datos de salida. Si aplicamos un 1 binario ($S_1 = 0$ y $S_0 = 1$), los datos de la entrada D_1 aparecerán en la salida de datos. Si se aplica un 2 binario ($S_1 = 1$ y $S_0 = 0$), obtendremos en la salida los datos de D_2. Si aplicamos un 3 binario ($S_1 = 1$ y $S_0 = 1$), los datos de D_3 serán conmutados a la línea de salida. El resumen del funcionamiento se puede ver en la Tabla 6.8.

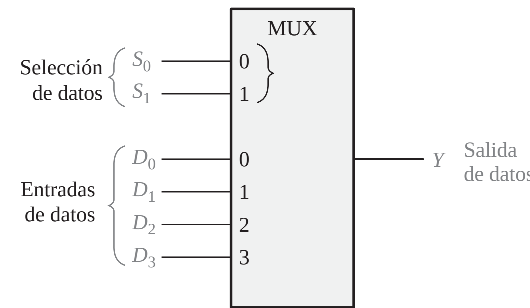


FIGURA 6.46 Símbolo lógico de un selector/multiplexor de datos de una salida y cuatro entradas.

Entradas de selección de datos		Entrada seleccionada
S_1	S_0	
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

TABLA 6.8 Selección de datos de un multiplexor de 1 salida y 4 entradas.

Ahora veamos la circuitería lógica necesaria para implementar esta operación de multiplexación. La salida de datos es igual al estado de la entrada de datos *seleccionada*. Por tanto, podemos deducir una expresión lógica para la salida en función de las entradas de datos y de las entradas de selección.

La salida de datos es igual a D_0 sólo si $S_1 = 0$ y $S_0 = 0 : Y = D_0\overline{S}_1\overline{S}_0$.

La salida de datos es igual a D_1 sólo si $S_1 = 0$ y $S_0 = 1 : Y = D_1\overline{S}_1S_0$.

La salida de datos es igual a D_2 sólo si $S_1 = 1$ y $S_0 = 0 : Y = D_2S_1\overline{S}_0$.

La salida de datos es igual a D_3 sólo si $S_1 = 1$ y $S_0 = 1 : Y = D_3S_1S_0$.

Si se aplica la operación OR a estos términos, la expresión total para la salida de datos es:

$$Y = D_0\overline{S}_1\overline{S}_0 + D_1\overline{S}_1S_0 + D_2S_1\overline{S}_0 + D_3S_1S_0$$

La implementación de esta ecuación requiere cuatro puertas AND de tres entradas, una puerta OR de cuatro entradas y dos inversores para generar los complementos de S_1 y S_0, como se muestra en la Figura 6.47. Dado que los datos pueden ser seleccionados desde cualquier línea de entrada, se conoce también a este circuito con el nombre de **selector de datos**.

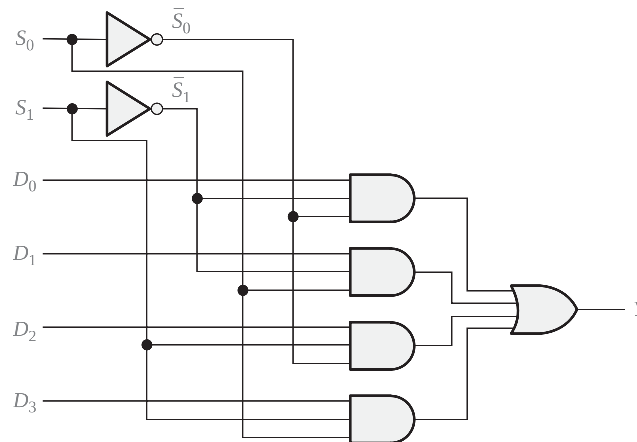
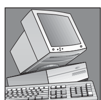


FIGURA 6.47 Diagrama lógico de un multiplexor de cuatro entradas.



NOTAS INFORMÁTICAS

Un *bus* es una ruta interna por la que se envían señales eléctricas desde una parte a otra de una computadora. En las redes de computadoras, un *bus compartido* es aquél que está conectado a todos los microprocesadores del sistema, con el fin de intercambiar datos. Un bus compartido puede contener dispositivos de memoria y de entrada/salida a los que pueden acceder todos los microprocesadores del sistema. El acceso al bus compartido se controla mediante un *árbitro de bus* (una especie de multiplexor), el cual hace que sólo un procesador utilice cada vez el bus compartido del sistema.

EJEMPLO 6.14

Se aplican las formas de onda de la Figura 6.47(a) a la entrada de datos y a la entrada de selección del multiplexor de la Figura 6.46. Determinar la señal de salida en relación a las entradas.

Solución El estado binario de las entradas de selección de datos durante cada intervalo determina cuál es la entrada de datos seleccionada. Observe que las entradas

selección de datos siguen la secuencia binaria repetitiva 00, 01, 10, 11, 00, 01, 10, 11, etc. La forma de onda de salida resultante se muestra en la Figura 6.48(b).

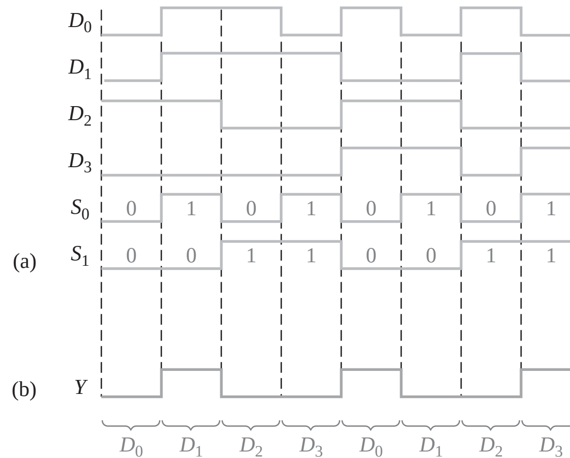
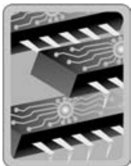


FIGURA 6.48

Problema relacionado Construir un diagrama de tiempos que muestre todas las entradas y la salida, si las formas de onda S_0 y S_1 de la Figura 6.48 se intercambian.

EL CUÁDRUPLE MULTIPLEXOR/SELECTOR DE DATOS DE 2 ENTRADAS 74HC157



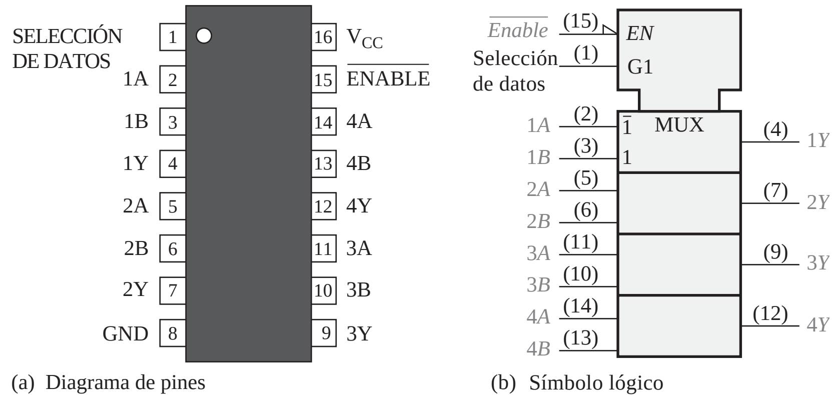
El 74HC157, al igual que su versión LS, está formado por cuatro multiplexores de dos entradas. Cada uno de los cuatro multiplexores comparten una misma línea de selección de datos y una de *habilitación* (*enable*). Ya que sólo existen dos entradas de datos que puedan ser seleccionadas en cada multiplexor, es suficiente con tener una única entrada de selección.

Un nivel BAJO en la entrada de *habilitación* ($\overline{Enable}$) permite al dato de entrada seleccionado pasar a la salida. Un nivel ALTO en la entrada $\overline{Enable}$ evita que los datos pasen a la salida, es decir, inhabilita los multiplexores. Este dispositivo puede estar disponible en otras familias CMOS o TTL. Consulte el sitio web de Texas Instruments en www.ti.com.

Símbolos lógicos ANSI/IEEE. En la Figura 6.49(a) se muestra el diagrama de pines del 74HC157 y su símbolo lógico ANSI/IEEE en la Figura 6.49(b). Observe que los cuatro multiplexores se representan mediante divisiones del bloque y que las entradas comunes a los cuatro multiplexores se indican como entradas al bloque recortado de la parte superior, que recibe el nombre de *bloque común de control*. Todas las etiquetas dentro del bloque superior del MUX se aplican a los bloques que haya por debajo.

Observe las etiquetas 1 y $\overline{1}$ de los bloques del MUX y la etiqueta G1 en el bloque común de control. Estas etiquetas son un ejemplo del sistema de **notación de dependencia** especificado en el estándar ANSI/IEEE 91-1984. En este caso, G1 indica una relación AND entre la entrada de selección de datos y las entradas de datos designadas por

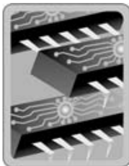
1 ó $\overline{1}$. El $\overline{1}$ indica que la relación AND se aplica al complemento de la entrada G1. En otras palabras, cuando la entrada de selección está a nivel ALTO, se seleccionan las entradas *B* de los multiplexores y, cuando la entrada de selección está a nivel BAJO, se seleccionan las entradas *A*. Para indicar dependencia AND siempre se usa una "G". Otros aspectos de la notación de dependencia serán tratados a lo largo del libro.



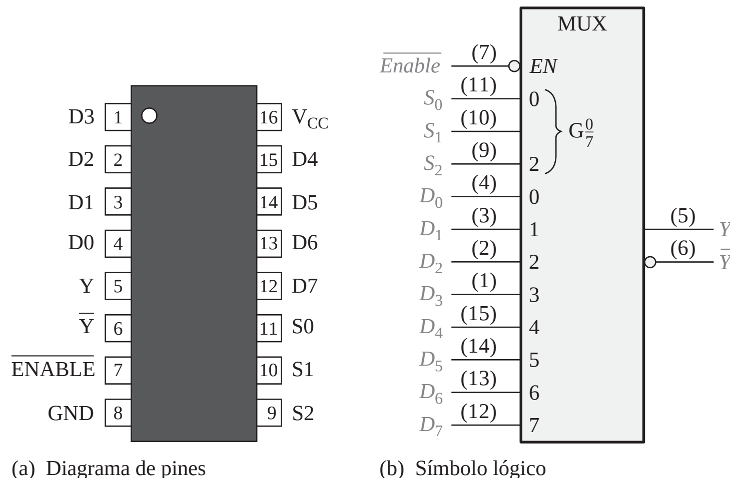
(a) Diagrama de pines (b) Símbolo lógico

FIGURA 6.49 Diagrama de pines y símbolo lógico para el cuádruple selector de datos/multiplexor de dos entradas 74HC157.

EL MULTIPLEXOR/SELECTOR DE DATOS DE 8 ENTRADAS 74LS151



El 74LS151 tiene ocho entradas de datos (D_0 –D_7) y, por tanto, tres líneas de entrada de dirección o de selección de datos (S_0-S_2). Se necesitan tres bits para seleccionar cualquiera de las ocho entradas de datos ($2^3 = 8$). Un nivel BAJO en la entrada de habilitación ($\overline{Enable}$) permite que los datos de entrada seleccionados pasen a la salida. Observe que se encuentran disponibles tanto la salida de datos como su complemento. En la Figura 6.50(a) se muestra el diagrama de pines y en la parte (b) el símbolo lógico ANSI/IEEE.



(a) Diagrama de pines (b) Símbolo lógico

FIGURA 6.50 Diagrama de pines y símbolo lógico para el multiplexor/selector de datos de 8 entradas 74LS151.

En este caso no hay necesidad de tener un bloque de control común en el símbolo lógico, ya que sólo hay que controlar un único multiplexor, y no cuatro como en el 74HC157. La etiqueta $G\frac{0}{7}$ dentro del símbolo lógico indica la relación AND entre las entradas de selección de datos y cada una de las entradas de datos, de la 0 a la 7. Este dispositivo puede estar disponible en otras familias CMOS o TTL. Consulte el sitio web de Texas Instruments en www.ti.com.

EJEMPLO 6.15

Utilizar multiplexores 74LS151 y cualquier otra lógica necesaria para multiplexar 16 líneas de datos en una única línea de salida de datos.

Solución

En la Figura 6.51 se muestra una implementación de este sistema. Se necesitan cuatro bits para seleccionar cualquiera de las 16 líneas de entrada de datos ($2^4 = 16$). En esta aplicación, la entrada de habilitación $(\overline{Enable})$ se utiliza como el bit más significativo de selección de datos. Cuando el MSB del código de selección de datos está a nivel BAJO, se habilita el 74LS151 de la izquierda y se selecciona una de las entradas de datos (D_0 a la D_7) mediante los otros tres bits de selección de datos. Cuando el MSB de selección de datos está a nivel ALTO, se habilita el 74LS151 de la derecha y se selecciona una de las entradas de datos (D_8 a la D_{15}). Los datos de entrada seleccionados pasan luego a través de la puerta negativa-OR y van a dar a la única línea de salida.

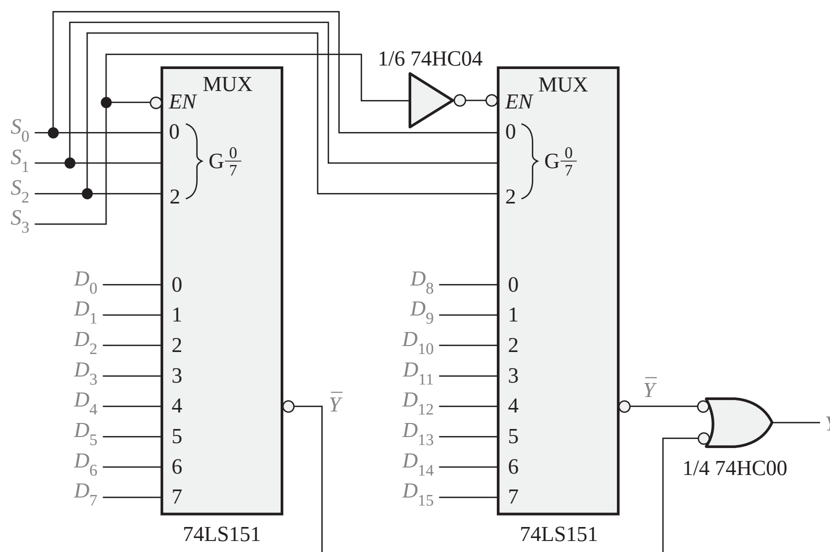


FIGURA 6.51 Multiplexor de 16 entradas.

Problema relacionado

Determinar los códigos necesarios en las entradas de selección de datos para seleccionar cada una de las siguientes entradas de datos: D_0, D_4, D_8 y D_{13}.

Aplicación

Display multiplexor de 7-segmentos. La Figura 6.52 muestra un método simplificado de multiplexación de números BCD para un display de 7-segmentos. En este ejemplo, se visualizan en el display de 7-segmentos números de dos dígitos, mediante el uso de un único decodificador BCD a 7-segmentos. Este método básico de multiplexación puede ampliarse para visualizar números con cualquier cantidad de dígitos.

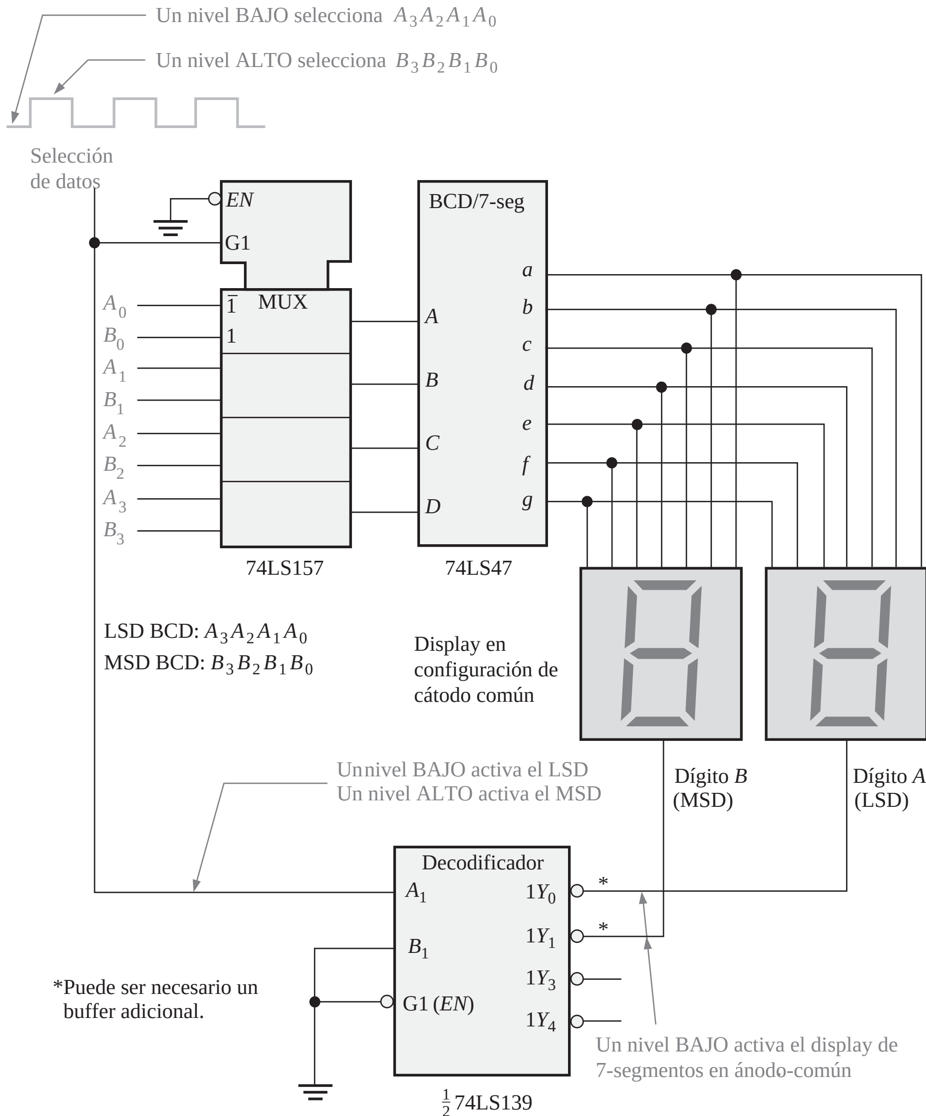


FIGURA 6.52 Lógica de multiplexación simplificada de un display de 7-segmentos.

Su funcionamiento básico es el siguiente. Se aplican dos dígitos BCD ($A_3A_2A_1A_0$ y $B_3B_2B_1B_0$) a las entradas de un multiplexor. Se aplica una señal cuadrada a la línea de selección de datos de forma que, cuando está a nivel BAJO, los bits de A ($A_3A_2A_1A_0$) pasan a las entradas del decodificador BCD a 7-segmentos74LS47. El

nivel BAJO en la entrada de selección de datos genera un nivel BAJO en la entrada A_1 del decodificador de 2-líneas a 4-líneas74LS139, activando su salida 0 y habilitando el display del dígito A, al conectar su terminal común a masa. El dígito A se encuentra ahora encendido, mientras que el B está apagado.

Cuando la línea de selección de datos pasa a nivel ALTO, los bits de B ($B_3B_2B_1B_0$) pasan a las entradas del decodificador BCD a 7-segmentos. Ahora se activa la salida 1 del decodificador 74LS139, encendiendo el display del dígito B, que pasa a visualizarse, mientras que el A se encuentra apagado. El ciclo se repite a la frecuencia de la señal cuadrada que se aplica a la entrada de selección de datos. Esta frecuencia tiene que ser lo suficientemente alta (unos 30 Hz) para evitar el parpadeo en los displays cuando se multiplexa la presentación de los dígitos.

Generador de funciones lógicas. Una aplicación muy útil de los multiplexores/selectores de datos consiste en la generación de funciones lógicas combinacionales en forma de suma de productos. Cuando se emplea de esta manera, este dispositivo puede reemplazar puertas lógicas discretas, puede reducir significativamente el número de circuitos integrados y permite que los cambios en el diseño sean mucho más sencillos.

Con el fin de ilustrar esto, se ha utilizado un multiplexor/selector de datos de 8 entradas 74LS151, para implementar cualquier función lógica de 3 variables, conectando las variables a las entradas de selección de datos y asignando a cada entrada de datos el nivel lógico requerido por la tabla de verdad para dicha función. Por ejemplo, si la función es 1 cuando la combinación de variables es $\overline{A}_2\overline{A}_1\overline{A}_0$, la entrada 2 (seleccionada por 010) se conecta a un nivel ALTO. Este nivel ALTO pasa a la salida cuando esta combinación particular de variables ocurre en las líneas de selección de datos. Un ejemplo nos servirá para clarificar esta aplicación.

EJEMPLO 6.16

Implementar la función lógica especificada en la Tabla 6.9, utilizando un multiplexor/selector de datos de 8 entradas 74LS151. Comparar este método con una implementación discreta con puertas lógicas.

Entradas			Salida
A_2	A_1	A_0	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

TABLA 6.9

Solución Observe en la tabla de verdad que Y vale 1 para las siguientes combinaciones de variables de entrada: 001, 011, 101 y 110. Para el resto de las combinaciones, Y vale 0. Para poder implementar esta función mediante el selector de datos, la entrada de datos seleccionada por cada una de las combinaciones mencionadas anteriormente tiene que conectarse a un nivel ALTO (5V). El resto de las entradas de datos debe conectarse a un nivel BAJO (tierra), como se muestra en la Figura 6.53.

La implementación de esta función mediante puertas lógicas requeriría cuatro puertas AND de 3 entradas, una puerta OR de 4 entradas y tres inversores, a menos que la expresión pudiera simplificarse.

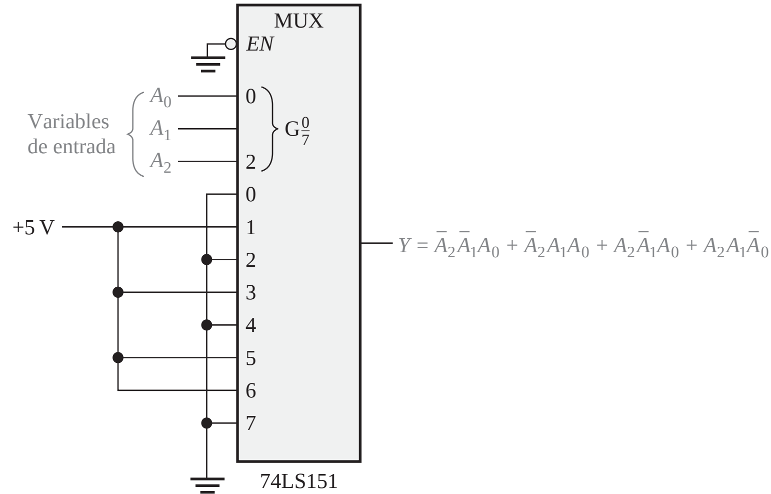


FIGURA 6.53 Multiplexor/selector de datos conectado como generador de funciones lógicas de 3 variables.

Problema relacionado Utilizar un 74LS151 para implementar la siguiente expresión:

$$Y = \overline{A}_2\overline{A}_1\overline{A}_0 + A_2\overline{A}_1\overline{A}_0 + \overline{A}_2A_1\overline{A}_0$$

El Ejemplo 6.16 ilustra cómo se puede utilizar el selector de datos de 8 entradas como generador de funciones lógicas de tres variables. En realidad, este dispositivo puede usarse también como generador de funciones lógicas de 4 variables utilizando uno de los bits (A_0) junto con las entradas de datos.

Una tabla de verdad de 4 variables da lugar a dieciséis combinaciones de las variables de entrada. Cuando se emplea un selector de datos de 8 bits, cada entrada se selecciona dos veces: la primera vez cuando A_0 es 0 y la segunda cuando A_0 es 1. Teniendo esto en cuenta. podemos aplicar las siguientes reglas (Y es la salida y A_0 es el bit menos significativo):

1. Si $Y = 0$ las dos veces que una entrada de datos dada se selecciona mediante una determinada combinación de las variables de entrada $A_3\ A_2\ A_1$, dicha entrada de datos se conecta a tierra (0).
2. Si $Y = 1$ las dos veces que una entrada de datos dada se selecciona mediante una determinada combinación de las variables de entrada $A_3\ A_2\ A_1$, dicha entrada de datos se conecta a $+V$ (1).
3. Si Y es diferente las dos veces que una entrada de datos dada se selecciona mediante una determinada combinación de las variables de entrada $A_3\ A_2\ A_1$, y si $Y = A_0$, la entrada de datos se conecta a A_0.
4. Si Y es diferente las dos veces que una entrada de datos dada se selecciona mediante una determinada combinación de las variables de entrada $A_3\ A_2\ A_1$, y si $Y = \overline{A}_0$ la entrada de datos se conecta a $\overline{A}_0$.

EJEMPLO 6.17

Implementar la función lógica de la Tabla 6.10 utilizando el multiplexor/selector de datos de 8 entradas 74LS151. Comparar este método con una implementación realizada con puertas lógicas discretas.

Dígito decimal	Entradas A_3	A_2	A_1	A_0	Salida Y
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	1
7	0	1	1	1	1
8	1	0	0	0	1
9	1	0	0	1	0
10	1	0	1	0	1
11	1	0	1	1	0
12	1	1	0	0	1
13	1	1	0	1	1
14	1	1	1	0	0
15	1	1	1	1	1

TABLA 6.10

Solución

Las entradas de selección de datos son $A_3A_2A_1$. En la primera fila de la tabla, $A_3A_2A_1 = 000$ e $Y = A_0$. En la segunda fila, de nuevo $A_3A_2A_1$ es 000, $Y = A_0$. Por

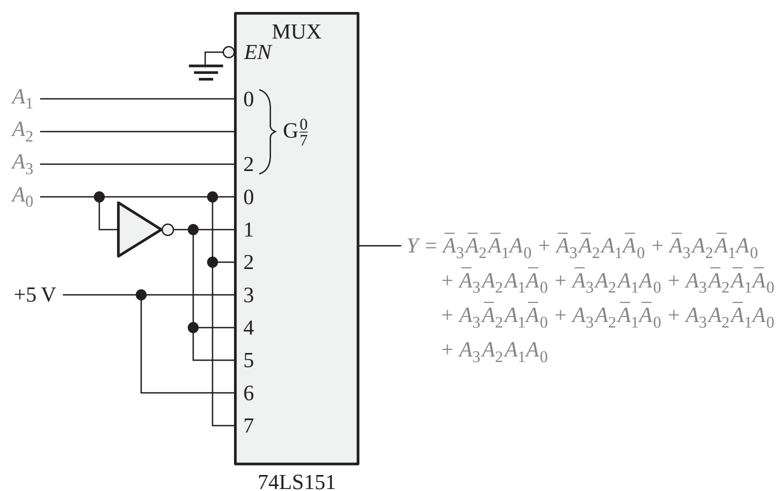


FIGURA 6.54 Multiplexor/selector de datos conectado como generador de funciones lógicas de 4 variables.

tanto, A_0 se conecta a la entrada 0. En la tercera fila de la tabla $A_3A_2A_1 = 001$, $Y = \overline{A}_0$. También, en la cuarta columna, $A_3A_2A_1$ es de nuevo 001 e $Y = \overline{A}_0$. Por tanto, A_0 se invierte y se conecta a la entrada 1. Este análisis continúa hasta que cada entrada se conecta apropiadamente de acuerdo con las reglas especificadas. La implementación se muestra en la Figura 6.54.

Si se implementara con puertas lógicas, la función requeriría al menos diez puertas AND de 4 entradas, una puerta OR de 10 entradas y cuatro inversores, aunque una posible simplificación reduciría este requisito.

Problema relacionado. Si, en la Tabla 6.10, $Y = 0$ cuando las entradas son todas cero y, alternativamente, 1 y 0 para las restantes filas de la tabla, implementar la función lógica resultante utilizando un 74LS151.

REVISIÓN DE LA SECCIÓN 6.8

1. En la Figura 6.47, $D_0 = 1$, $D_1 = 0$, $D_2 = 1$, $D_3 = 0$, $S_0 = 1$ y $S_1 = 0$. ¿Cuál es la salida?
2. Identificar cada dispositivo:
   (a) 74LS157 **(b)** 74LS151
3. En las entradas de datos de un 74LS151 se aplican alternativamente niveles BAJOS y ALTOS, comenzando por $D_0 = 0$. Las líneas de selección de datos se secuencian mediante un contador binario (000, 001, 010, etc.) a una frecuencia de 1 kHz. La entrada de habilitación está a nivel BAJO. Describir la forma de onda de salida.
4. Describir brevemente el propósito de cada uno de los siguientes dispositivos de la Figura 6.52.
   (a) 74LS157 **(b)** 74LS47 **(c)** 74LS139

6.9 DEMULTIPLEXORES

Un ***demultiplexor*** (DEMUX) básicamente realiza la función contraria a la del multiplexor. Toma datos de una línea y los distribuye a un determinado número de líneas de salida. Por este motivo, el demultiplexor se conoce también como distribuidor de datos. Como veremos, los decodificadores pueden utilizarse también como demultiplexores.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Explicar el funcionamiento básico de un demultiplexor. ■ Describir cómo el decodificador de 4-líneas a 16-líneas 74HC154 puede utilizarse como demultiplexor. ■ Desarrollar el diagrama de tiempos de un demultiplexor con un número determinado de entradas de datos y de selección de datos.

▲ *En un demultiplexor, los datos pasan de una línea a varias líneas.*

La Figura 6.55 muestra un circuito demultiplexor (DEMUX) de 1-línea a 4-líneas. La línea de entrada de datos está conectada a todas las puertas AND. Las dos líneas de selección de datos activan únicamente una puerta cada vez y los datos que aparecen en la línea de entrada de datos pasarán a través de la puerta seleccionada hasta la línea de salida de datos asociada.

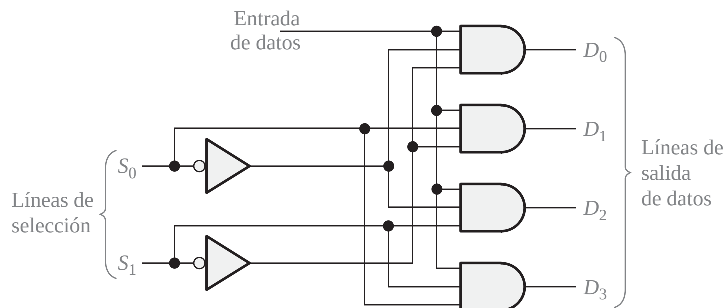


FIGURA 6.55 Demultiplexor de 1-línea a 4-líneas

EJEMPLO 6.18

En la Figura 6.56 se muestra una forma de onda de entrada de datos serie y las entradas de selección de datos (S_0 y S_1). Determinar las formas de onda de datos de salida que obtendríamos en las salidas D_0 hasta la D_3 para el demultiplexor de la Figura 6.55.

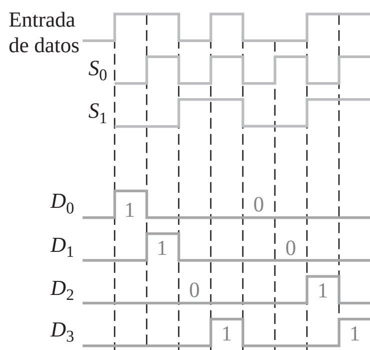
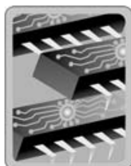


FIGURA 6.56

Solución Las líneas de selección de datos reciben una secuencia binaria que hace que cada bit de entrada sucesivo sea redirigido hacia D_0, D_1, D_2 y D_3 secuencialmente, como se puede ver en las formas de onda de salida de la Figura 6.56.

Problema relacionado Desarrollar el diagrama de tiempos del demultiplexor si se invierten las señales S_0 y S_1.

EL DEMULTIPLEXOR 74HC154



Hasta ahora hemos visto el 74HC154 como decodificador de 4-líneas a 16-líneas (Sección 6.5). Este dispositivo, así como otros decodificadores, se utiliza también en diversas aplicaciones como demultiplexor. El símbolo lógico de este circuito, cuando se emplea como demultiplexor, se muestra en la Figura 6.57. Cuando se utiliza con este fin, se usan las líneas de entrada como líneas de selección de datos, una de las entradas de

activación del chip se usa como línea de entrada de datos y la otra se mantiene a nivel BAJO, para activar la puerta interna negativa-AND que se encuentra en la parte inferior del diagrama. Este dispositivo puede estar disponible en otras familias CMOS o TTL. Consulte el sitio web de Texas Instruments en www.ti.com.

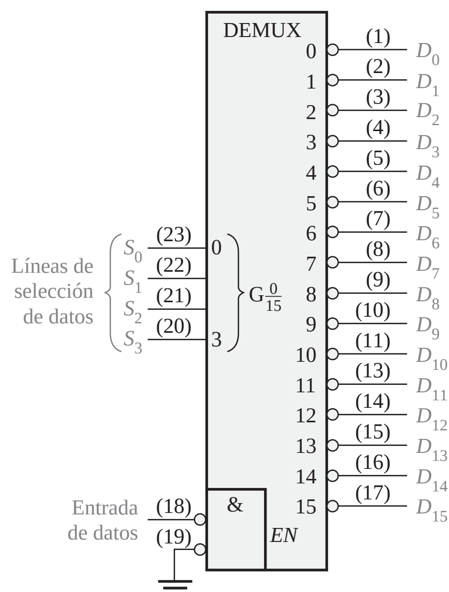


FIGURA 6.57 El decodificador 74HC154 utilizado como demultiplexor.

REVISIÓN DE LA SECCIÓN 6.9

1. En general, ¿cómo puede utilizarse un decodificador como demultiplexor?
2. El demultiplexor 74HC154 de la Figura 6.57 tiene en las líneas de selección de datos el código binario 1010 y la línea de entrada de datos está a nivel BAJO. ¿Cuáles son los estados de las líneas de salida?

6.10 GENERADORES / COMPROBADORES DE PARIDAD

Cuando se transfieren datos digitales de un punto a otro dentro de un sistema digital o cuando se transmiten códigos desde un sistema a otro, se pueden producir errores. Estos errores se manifiestan mediante cambios indeseados en los bits que conforman la información codificada; es decir, un 1 puede cambiar a 0 o un 0 a 1, debido a un mal funcionamiento de los componentes o al ruido eléctrico. En la mayoría de los sistemas digitales, la probabilidad de que haya un bit erróneo es muy pequeña, y la de que haya más de uno es todavía menor. En cualquier caso, cuando no se detecta un error, pueden originarse serios problemas en un sistema digital.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Explicar el concepto de paridad. ■ Implementar un circuito de paridad básico con puertas OR-exclusiva. ■ Describir el funcionamiento de la lógica de un generador/comprobador de paridad básico. ■ Explicar el generador/comprobador de paridad de 9 bits 74LS280. ■ Explicar cómo se puede implementar la detección de errores en una transmisión de datos.

En el Capítulo 2 se ha estudiado el método de paridad de detección de errores, en el que se añade un ***bit de paridad*** a un grupo de bits de información para conseguir que el número total de 1s sea par o impar (dependiendo del sistema que se trate). Además de los bits de paridad, hay otros códigos específicos que también permiten realizar la detección de errores.

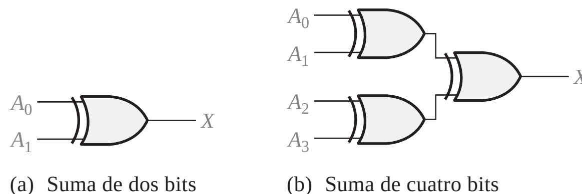
Lógica básica de la paridad

▲ *Un bit de paridad indica si el número de 1s en un código es par o impar con el fin de detectar errores.*

Para poder comprobar o generar la paridad adecuada dentro de un determinado código, se puede aplicar un principio muy sencillo:

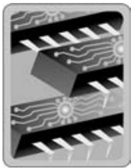
La suma (descartando los acarreos) de un número par de 1s siempre es 0 y la suma de un número impar de 1s siempre es 1.

Por tanto, para determinar si un cierto código tiene **paridad par** o **paridad impar**, se suman todos los bits de ese código. Como sabemos, la suma de dos bits se puede generar mediante una puerta OR-exclusiva, como se muestra en la Figura 6.58(a); la suma de cuatro bits se puede realizar a partir de tres puertas OR-exclusiva conectadas como se indica en la Figura 6.58(b), y así sucesivamente. Cuando el número de 1s en las entradas es par, la salida X es 0 (nivel BAJO). Cuando el número de 1s es impar, la salida X es 1 (nivel ALTO).



(a) Suma de dos bits (b) Suma de cuatro bits **FIGURA 6.58**

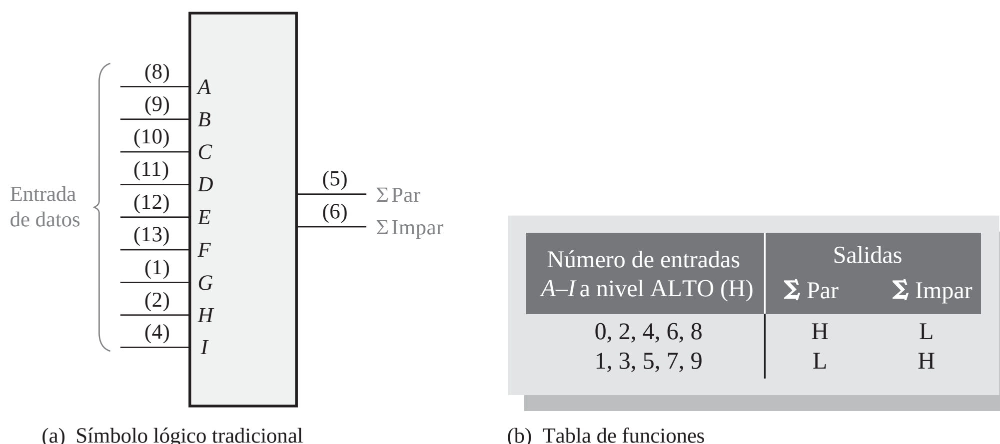
EL GENERADOR/COMPROBADOR DE PARIDAD DE 9 BITS 74LS280



El símbolo lógico y la tabla de funciones de un 74LS280 se representa en la Figura 6.59. Este dispositivo se puede utilizar para comprobar la paridad par o impar en un código de 9 bits (ocho bits de datos y un bit de paridad), o puede también emplearse para generar un bit de paridad para un código binario de hasta 9 bits. Sus entradas son desde A hasta I; cuando en las entradas hay un número par de 1s, la salida ΣPar es un nivel ALTO y la salida ΣImpar es un nivel BAJO. Este dispositivo puede estar disponible en otras familias CMOS o TTL. Consulte el sitio web de Texas Instruments en www.ti.com.

Comprobador de paridad. Cuando este dispositivo se utiliza como un comprobador de paridad par, el número de bits de entrada deberá ser siempre par; y cuando se produzca un error, la salida ΣPar pasará a nivel BAJO (L) y la salida ΣImpar será un nivel ALTO (H). Cuando se emplea como comprobador de paridad impar, el número de bits de entrada deberá ser siempre impar, y cuando se produzca un error, la salida ΣImpar será un nivel BAJO (L) y la salida ΣPar será un nivel ALTO (H).

Generador de paridad. Si este dispositivo se utiliza como generador de paridad par, el bit de paridad se toma en la salida ΣImpar, ya que esta salida es 0 cuando hay un número par de bits de entrada y 1 cuando hay un número impar. Cuando se emplea como generador de paridad impar, el bit de paridad se toma en la salida ΣPar, dado que ésta es 0 cuando el número de bits de entrada es impar.



Número de entradas *A*–*I* a nivel ALTO (H)	Salidas	
	Σ Par	Σ Impar
0, 2, 4, 6, 8	H	L
1, 3, 5, 7, 9	L	H

(a) Símbolo lógico tradicional (b) Tabla de funciones

FIGURA 6.59 El generador/comprobador de paridad de 9 bits 74LS280

Sistema de transmisión de datos con detección de errores

En la Figura 6.60 se muestra un sistema simplificado de transmisión de datos para ilustrar una aplicación de los generadores/comprobadores de paridad, así como de los multiplexores y demultiplexores, y para ilustrar la necesidad de dispositivos de almacenamiento de datos en algunas aplicaciones.

En esta aplicación, los datos digitales procedentes de siete fuentes se multiplexan en una única línea para ser transmitidos a un punto distante. Se aplican los siete bits de datos (D_0 hasta D_6) a las entradas de datos del multiplexor y, al mismo tiempo, a las entradas del generador de paridad par. La salida ΣImpar del generador de paridad se utiliza como bit de paridad par. Este bit es 0 si el número de 1s en las entradas de la *A* a la *I* es par, y es 1 si el número de 1s en las mismas entradas es impar. Éste es el bit D_7 del código transmitido.

Las entradas de selección de datos van pasando cíclicamente por una secuencia binaria y cada bit de datos, comenzando en D_0, se transmite en serie por la línea de transmisión ($\overline{Y}$). En este ejemplo, la línea de transmisión está formada por cuatro conductores: uno para los datos serie y los otros tres para las señales de temporización (selección de datos). Existen maneras más sofisticadas de enviar información de temporización, pero estamos usando este método directo para ilustrar un principio básico.



NOTAS INFORMÁTICAS

El microprocesador Pentium realiza comprobaciones internas de paridad, así como comprobaciones de paridad de los buses externos de direcciones y datos. En una operación de lectura, el sistema externo puede transferir la información de paridad junto con los bytes de datos. El microprocesador Pentium comprueba si la paridad resultante es par y envía la señal correspondiente. Cuando se envía un código de dirección, este microprocesador no lleva a cabo una comprobación de paridad de la dirección, pero sí que genera un bit de paridad para la dirección.

En el extremo demultiplexor del sistema, las señales de selección de datos y la cadena de datos serie se aplican al demultiplexor. Los bits de datos se distribuyen mediante el demultiplexor a las líneas de salida en el orden en que llegaron a las entradas del multiplexor. Es decir, D_0 llega a la salida D_0, D_1 llega a la salida D_1, etc. El bit de paridad llega a la salida D_7. Estos ocho bits se almacenan temporalmente y se aplican al comprobador de paridad par. No todos estos bits se encuentran presentes en las entradas del comprobador de paridad hasta que el bit de paridad D_7 aparece y se almacena. En este instante, la puerta de error es activada por el código de selección de datos 111. Si la paridad es correcta, aparece un 0 en la salida ΣPar, manteniendo la

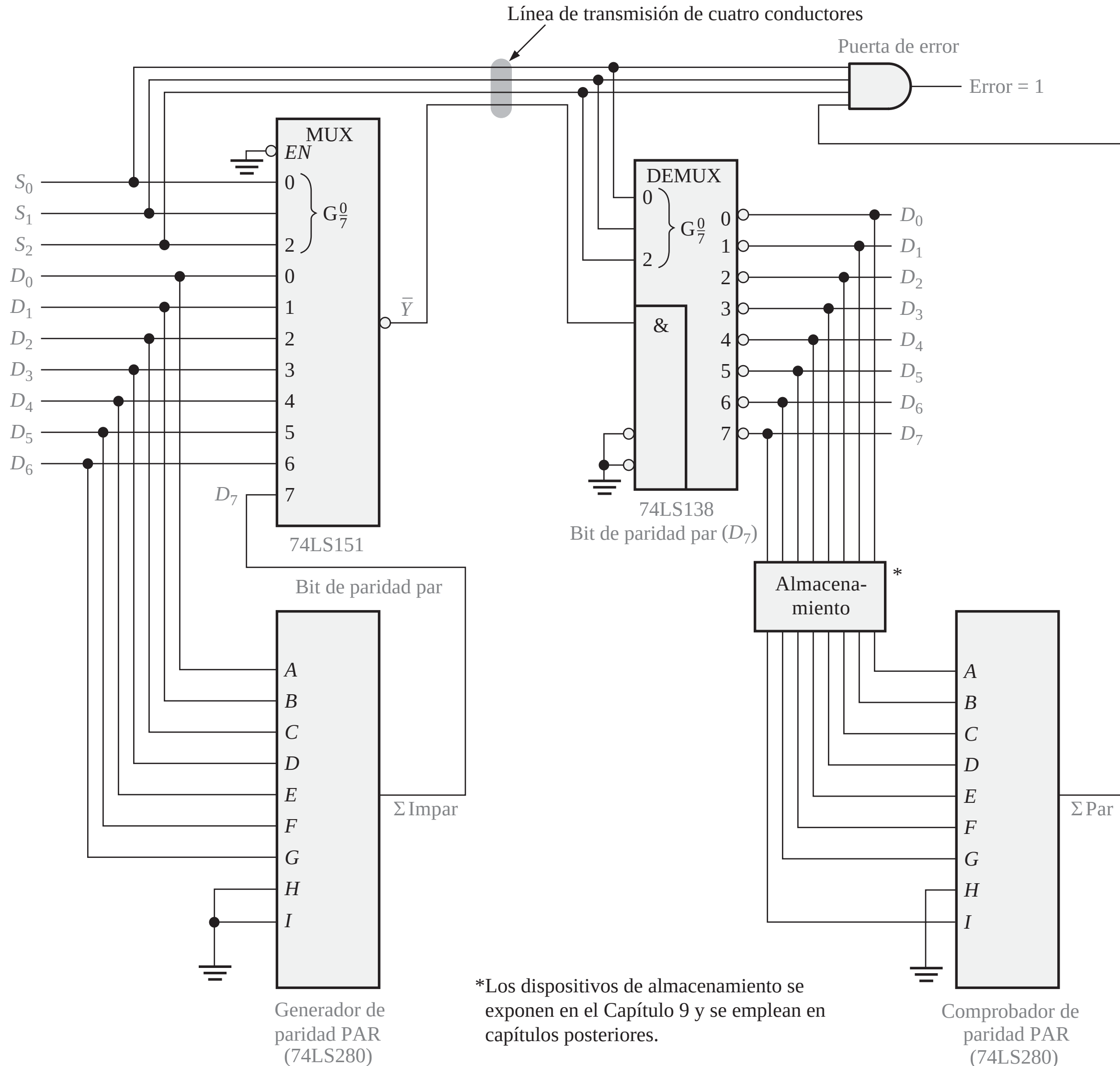


FIGURA 6.60 Sistema simplificado de transmisión de datos con detección de errores.

salida ERROR a nivel 0. Si la paridad es incorrecta, todos los 1s aparecerán en las entradas de la puerta de error, lo que da lugar a un 1 en la salida ERROR.

Esta aplicación particular demuestra que es necesario disponer de algún dispositivo de almacenamiento de datos, por lo que, cuando en el Capítulo 7 estudiemos los dispositivos de almacenamiento y los utilicemos en capítulos posteriores, seremos capaces de apreciar mejor su utilidad.

El diagrama de tiempos que se muestra en la Figura 6.61 ilustra el caso específico de transmisión de dos palabras de 8 bits, una de las cuales tiene paridad correcta y la otra se transmite con un error.

REVISIÓN DE LA SECCIÓN 6.10

1. Añadir un bit de paridad par a cada uno de los siguientes códigos:
(a) 110100 **(b)** 01100011

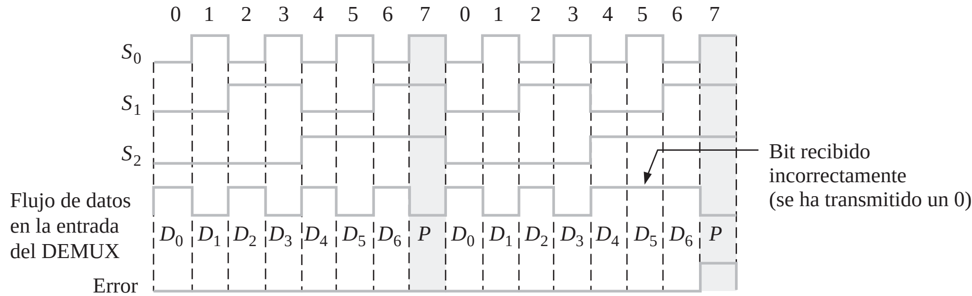


FIGURA 6.61 Ejemplo de transmisión de datos con y sin error en el sistema de la Figura 6.60.

2. Añadir un bit de paridad impar a cada uno de los siguientes códigos:

(a) 1010101 **(b)** 1000001

3. Comprobar cada uno de los siguientes códigos con paridad par e indicar si contienen errores.

(a) 100010101 **(b)** 1110111001

6.11 LOCALIZACIÓN DE AVERÍAS

En esta sección, se introduce y examina el problema de los impulsos de muy corta duración (*glitches*) en los decodificadores desde un punto de vista práctico. Un ***glitch*** es un pico de tensión o de corriente (impulso) no deseado de muy corta duración. Los circuitos lógicos pueden interpretar estos impulsos como una señal válida, originando fallos en el funcionamiento del circuito.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Explicar qué es un glitch. ■ Determinar la causa de los *glitches* en una aplicación de un decodificador. ■ Utilizar el método de la validación de salida para eliminar *glitches*.

El circuito 74LS138 se ha utilizado como demultiplexor en el sistema de transmisión de datos de la Figura 6.60. Ahora, en la Figura 6.62, se utiliza como un decodificador de 3-líneas a 8-líneas (binario-octal), para ilustrar cómo pueden ocurrir los *glitches* y cómo identificar su origen. Las entradas $A_2A_1A_0$ del decodificador se secuencian mediante un contador binario y las señales resultantes de entrada y salida se pueden visualizar en la pantalla de un analizador lógico, como se muestra en la Figura 6.62. Las transiciones de la señal A_2 están retrasadas con respecto a las transiciones de A_1, y las de A_1 respecto a las transiciones de A_0. Esto es lo que suele ocurrir cuando se emplea un contador binario para generar las señales, como veremos en el Capítulo 8.

Las señales de salida son correctas excepto por los *glitches* que aparecen en algunas de ellas. Se puede utilizar un osciloscopio o un analizador lógico para visualizar los *glitches*, que normalmente son difíciles de ver. Generalmente, es preferible el analizador lógico, especialmente para detectar las velocidades de repetición bajas (menores de 10 kHz) y/o su ocurrencia irregular, ya que la mayoría de los analizadores lógicos disponen de la función de *captura de glitches*. Los osciloscopios se pueden emplear para observar los *glitches* con cierta seguridad, especialmente si éstos se producen con una velocidad de repetición alta y constante (mayor que 10 kHz).

Los puntos de interés, que son las zonas marcadas en las señales de entrada de la Figura 6.62, se visualizan como se muestra en la Figura 6.63. En el punto 1, se produce una transición por el estado 000 debido a las diferencias de los retardos de las señales. Esto origina el primer *glitch* en la salida $\overline{0}$ del decodificador. En el punto 2, aparecen dos estados de transición, 010 y 000. Estos originan un *glitch* en la salida $\overline{2}$ del decodi-

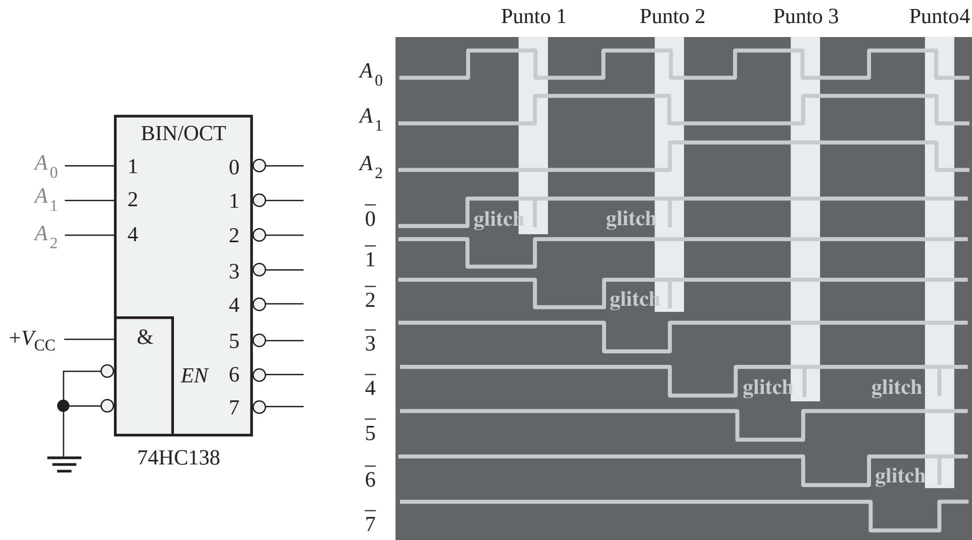


FIGURA 6.62 Formas de onda del decodificador con *glitches* en la salida.

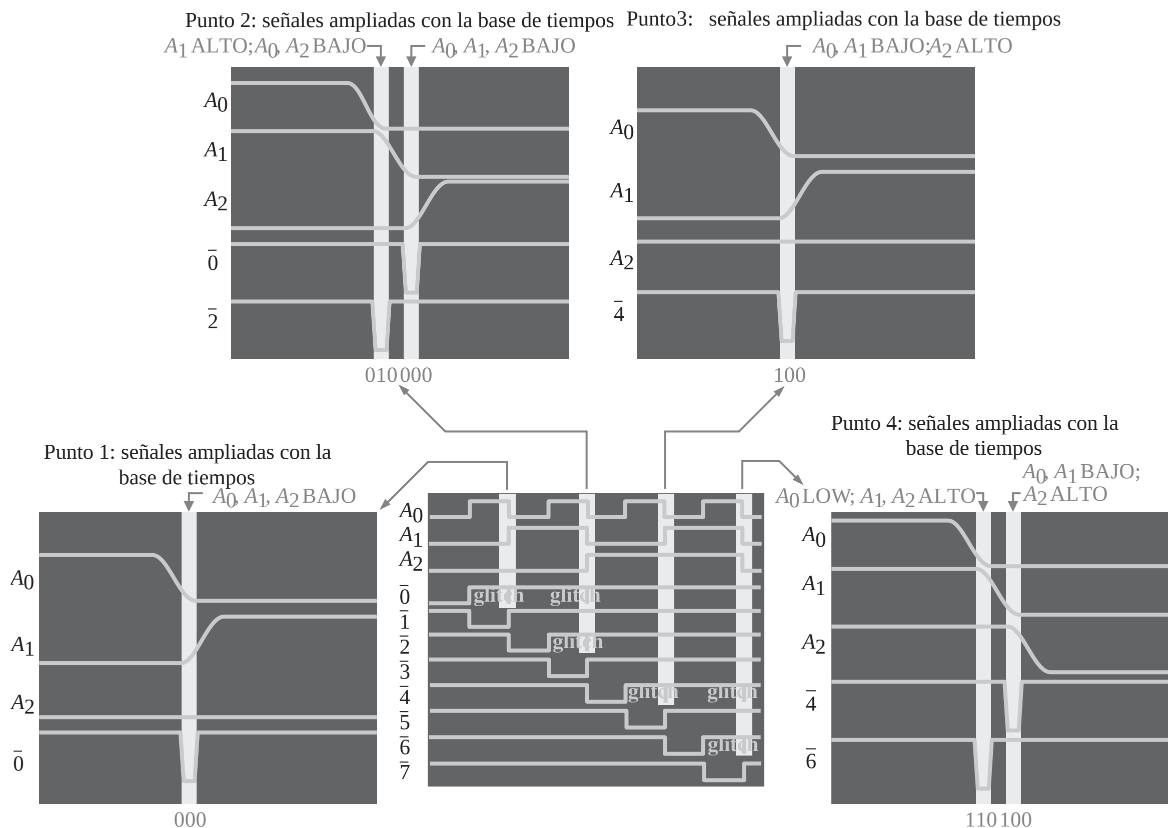


FIGURA 6.63 Formas de onda del decodificador que muestran cómo los estados de transición de entrada producen *glitches* en las señales de salida.

ficador y un segundo *glitch* en la salida $\overline{0}$, respectivamente. En el punto 3, el estado de transición es 100, lo que origina el primer *glitch* en la salida $\overline{4}$ del decodificador. En el punto 4, los dos estados de transición son 110 y 100, los cuales originan el *glitch* en la salida $\overline{6}$ del decodificador y en la salida $\overline{4}$, respectivamente.

Una manera de eliminar este problema es aplicar impulsos de **validación** (*strobing*), lo que consiste en activar el decodificador mediante un impulso de validación (*strobe*) únicamente durante los intervalos de tiempo en que las señales no se encuentran en un estado de transición. Este método se ilustra en la Figura 6.64.

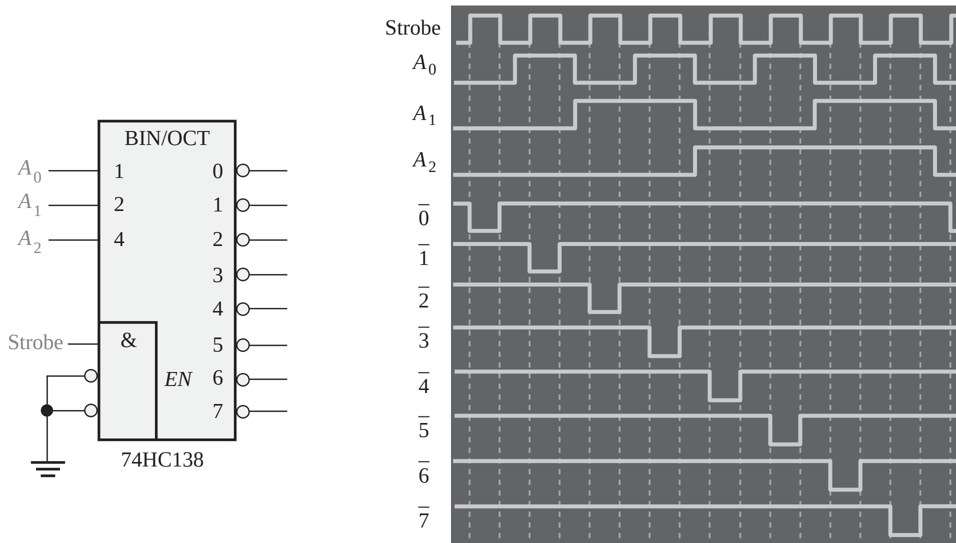


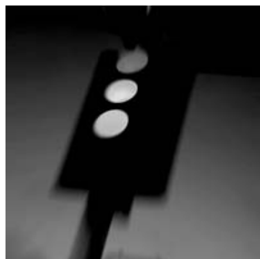
FIGURA 6.6 Aplicación de una señal de validación (*strobe*) para eliminar *glitches* en las salidas del decodificador.

CONSEJOS PRÁCTICOS

Además de los *glitches* que son el resultado de los retardos de propagación, como hemos visto en el caso de un decodificador, existen otros tipos de impulsos de ruido no deseados que pueden constituir también un problema. Los impulsos de corriente y tensión en las líneas de masa y alimentación (V_{CC}) son debidos a las señales de conmutación rápida en los circuitos digitales. Este problema se puede minimizar realizando un apropiado diseño de la placa de circuito impreso. Los impulsos de conmutación pueden ser absorbidos mediante el desacoplo de la tarjeta de circuito impreso con condensadores de 1 μF entre V_{CC} y masa. También deberían distribuirse condensadores más pequeños de desacoplo (0,022 μF a 0,1 μF) en distintos puntos de la placa de circuito impreso entre V_{CC} y masa. El desacoplo debería realizarse muy próximo a los dispositivos que conmutan a altas velocidades o que excitan más cargas, como por ejemplo, osciladores, contadores, buffers y controladores de bus.

REVISIÓN DE LA SECCIÓN 6.11

1. Definir el término *glitch*.
2. Explicar la principal causa de los *glitches* en los decodificadores.
3. Definir el impulso de *validación* (*strobe*).



APLICACIÓN A LOS SISTEMAS DIGITALES

En esta aplicación, vamos a comenzar a trabajar con el sistema de control de semáforos. En esta sección se establecen los requisitos del sistema, se desarrolla un diagrama de bloques, así como un diagrama de estados para ayudar a establecer la secuencia de funcionamiento. Diseñaremos la parte del sistema que involucra lógica combinacional y se propondrán los métodos de prueba. En los Capítulos 7 y 8 se tratarán los circuitos de lógica secuencial y de temporización del sistema.

Requerimientos generales del sistema

Se requiere un controlador digital para controlar un semáforo en la intersección de una calle de tráfico muy denso con una calle de tráfico moderado. La calle principal va a tener una luz verde durante un mínimo de 25 seg. o mientras no haya ningún vehículo en la calle perpendicular. Esta calle lateral tiene que tener la luz verde hasta que no circule ningún coche por ella, o durante un máximo de 25 seg. La luz ámbar de precaución tiene que durar 4 seg. en los cambios de luz verde a roja en ambas calles, principal y lateral. Estos requisitos se muestran en el diagrama de la Figura 6.65.

Desarrollo de un diagrama de bloques del sistema

A partir de los requisitos, se puede desarrollar un diagrama de bloques del sistema. En primer lugar, sabemos que el sistema tiene que controlar seis pares de luces diferentes. Estas son las luces roja, ámbar y verde para ambos sentidos, tanto en la calle principal como en la lateral. También sabemos que existe una entrada externa (además de la alimentación) que proviene de un sensor de vehículos situado en la calle lateral. En la Figura 6.66, puede ver un diagrama de bloques mínimo que ilustra estos requisitos.

A partir del diagrama de bloques mínimo vamos a ir entrando en los detalles. El sistema tiene cuatro estados, como se indica en la Figura 6.65, por lo que se necesita un circuito lógico para controlar la secuencia de estados (lógica secuencial). Además, se necesitan circuitos para generar los intervalos de tiempo adecuados de 25 seg. y 4 seg., que se requieren en el sistema y para generar una señal de reloj cíclica en el sistema (circuitos de temporización). Los intervalos de tiempo (largo y corto) y el sensor de vehículos son entradas de la lógica secuencial, dado que la secuenciación de estados es una función de estas variables. Se necesitan también circuitos lógicos para determinar cuál de los cuatro estados del sistema está activo en un determinado instante de tiempo, para así generar las sali-

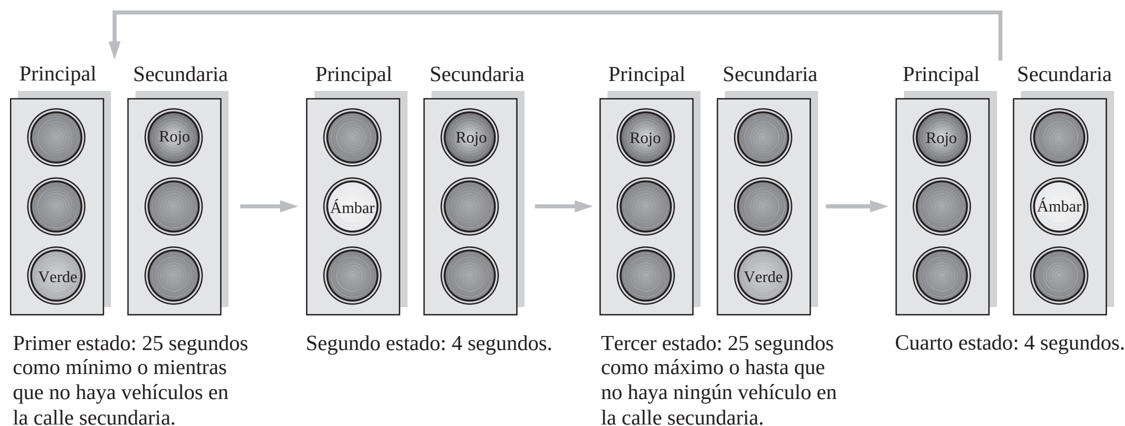


FIGURA 6.65 Requisitos para la secuencia de luces de los semáforos.

das adecuadas en las luces (decodificación de estados y lógica de salida), y para iniciar los intervalos de tiempo largo y corto. Finalmente, se necesita un circuito de interfaz para convertir los niveles lógicos de la decodificación y del circuito de salida en las tensiones y corrientes requeridas para encender cada una de las luces. La Figura 6.67 representa un diagrama de bloques más detallado que muestra estos elementos esenciales.

El diagrama de estados

Un diagrama de estados nos muestra gráficamente la secuencia de estados en un sistema y las condiciones de cada estado y de las transiciones entre cada uno de ellos. En realidad, la Figura 6.65 es, en cierta medida, un diagrama de estados, ya que muestra la secuencia de estados y las distintas condiciones.

Definición de las variables. Antes de poder desarrollar un diagrama de estados tradicional, es necesario definir las variables que determinan cómo pasa el sistema a través de los diferentes estados. A continuación se enumeran estas variables y sus símbolos:

- Presencia de vehículos en la calle lateral = V_s
- El temporizador de 25 s. (largo) está *activado* = T_L
- El temporizador de 4 s. (corto) está *activado* = T_S

El uso de variables complementadas indica la condición contraria. Por ejemplo, $\overline{V}_s$ indica que no hay ningún vehículo en la calle lateral; $\overline{T}_L$ indica que el temporizador de larga duración está desactivado y $\overline{T}_S$ indica que el temporizador de corta duración está desactivado.

Descripción del diagrama de estados. En la Figura 6.68 se muestra un diagrama de estados. Cada uno de los cuatro estados se etiqueta de acuerdo a la secuencia de 2 bits en código Gray, como se indica mediante los círculos. La flecha circular en cada estado indica que el sistema permanece en dicho estado bajo la condición definida por la variable o expresión asociada. Cada una de las flechas que van de un estado al siguiente indican un cambio de estado cuando se produce la condición definida por la variable o expresión asociada.

Primer estado El código Gray para este estado es 00. El semáforo de la calle principal está en verde y el de la calle lateral está en rojo. El sistema permanece en este estado al menos 25 segundos cuando el temporizador largo se encuentra activado o mientras que no haya ningún vehículo en la calle lateral $(T_L + \overline{V}_s)$. El sistema pasa al siguiente estado cuando el temporizador de 25 segundos está desactivado o cuando aparece algún vehículo en la calle secundaria $(\overline{T}_L V_s)$.

Segundo estado El código Gray para este estado es 01. El semáforo de la calle principal está en ámbar (precaución) y el de la calle lateral está en rojo. El sistema permanece en este estado durante 4 segundos mientras el temporizador corto está activado (T_S) y pasa al siguiente estado cuando este temporizador se desactiva $(\overline{T}_S)$.

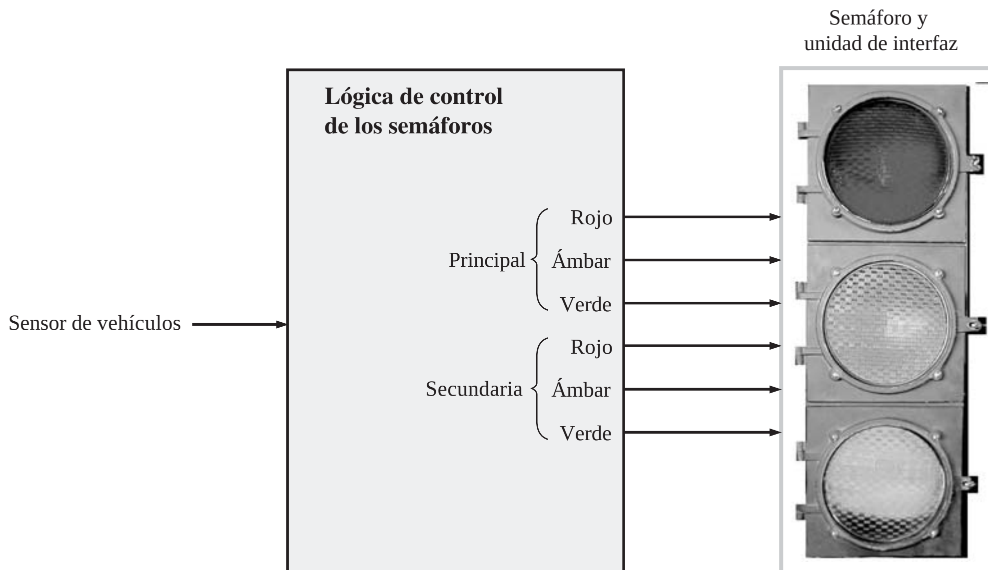


FIGURA 6.66 Diagrama de bloques mínimo del sistema.

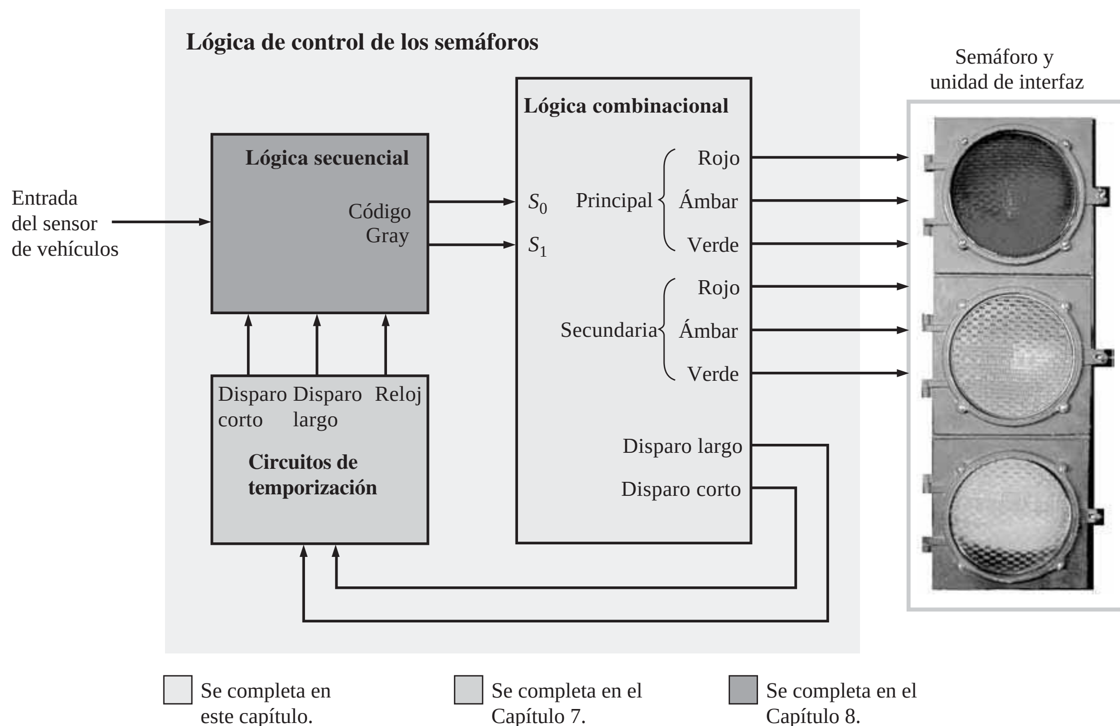


FIGURA 6.67 Diagrama de bloques del sistema en el que se indican los elementos esenciales.

Tercer estado El código Gray para este estado es 11. El semáforo de la calle principal está en rojo y el de la calle lateral está en verde. El sistema permanece en este estado cuando el temporizador largo se encuentra activado y hay un vehículo en la calle lateral ($T_L V_S$). El sistema pasa al siguiente estado cuando han transcurrido los 25 segundos o cuando no hay ningún vehículo en la calle secundaria, lo primero que ocurra $(\overline{T}_L + \overline{V}_s)$.

Cuarto estado El código Gray para este estado es 10. El semáforo de la calle principal está en rojo y el de la calle lateral está en ámbar. El sistema permanece en este estado 4 segundos cuando el temporizador corto se encuentra activado (T_S) y vuelve al primer estado cuando el temporizador corto se desactiva $(\overline{T}_S)$.

La lógica combinacional

En esta sección dedicada a las aplicaciones de sistemas nos vamos a centrar en la parte correspondiente a la lógica combinacional del diagrama de bloques de la Figura 6.67. Los circuitos de temporización y de la lógica secuencial serán el tema del que tratemos en las secciones dedicadas a las aplicaciones de sistemas de los Capítulos 7 y 8.

El primer paso en el diseño de la lógica consiste en desarrollar un diagrama de bloques para la parte de la lógica combinacional del sistema. Las tres funciones que esta lógica debe realizar se definen a continuación y el diagrama resultante, con un bloque para cada una de las tres funciones, se muestra en la Figura 6.69:

- **Decodificador de estados.** Decodificar el código Gray de 2 bits de la lógica secuencial, para determinar en cuál de los cuatro estados se encuentra el sistema.
- **Lógica de salida de las luces.** Utilizar el estado decodificado para activar (mediante los circuitos de interfaz) las dos luces de los semáforos apropiadas, de las seis luces existentes.
- **Lógica del circuito de disparo.** Utilizar los estados decodificados para generar las señales que inicialicen (disparen) adecuadamente los temporizadores largo y corto.

Implementación de la lógica combinacional

Implementación de la lógica del decodificador El decodificador de estados tiene dos entradas (código Gray de 2

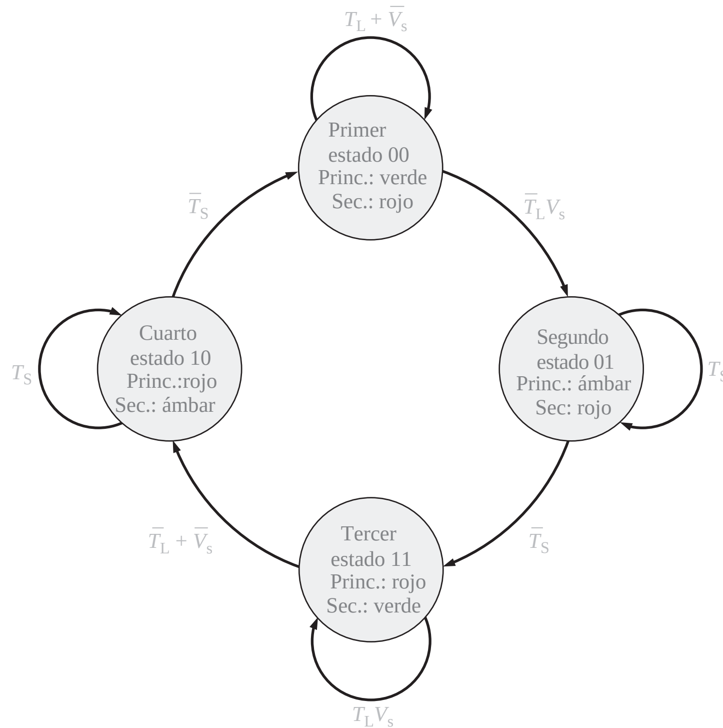


FIGURA 6.68 Diagrama de estados del sistema de control de semáforos, indicando la secuencia de código Gray.

bits) y tiene que tener una salida para cada uno de los cuatro estados, como se muestra en la Figura 6.70. Las dos entradas en código Gray se designan por S_0 y S_1 y las cuatro salidas de estado se etiquetan como SO_1, SO_2, SO_3 y SO_4. Las expresiones booleanas para las salidas de los estados son las siguientes:

$$SO_1 = \overline{S}_1\overline{S}_0$$

$$SO_2 = \overline{S}_1 S_0$$

$$SO_3 = S_1 S_0$$

$$SO_4 = S_1\overline{S}_0$$

La tabla de verdad para la lógica del decodificador de estados se muestra en la Tabla 6.11.

Implementación de la lógica de salida de las luces Esta lógica tiene que tomar las cuatro salidas de estado y generar seis salidas para activar las luces de los semáforos. Estas salidas se designan por *MR, MY, MG* (para la luces roja, ámbar y verde del semáforo de la calle principal) y por *SR, SY, SG* (para la luces roja, ámbar y verde del semáforo de la calle secundaria). Usando como referencia la Tabla 6.11, vemos que las salidas de las luces de los semáforos puede expresarse como:

$$MR = SO_3 + SO_4 \qquad SR = SO_1 + SO_2$$

$$MY = SO_2 \qquad SY = SO_4$$

$$MG = SO_1 \qquad SG = SO_3$$

La lógica de salida se implementa como se muestra en la Figura 6.71.

Implementación de la lógica del circuito de disparo La lógica de disparo produce dos salidas. La salida para el circuito de temporización de 25 s. (temporizador largo) da lugar a una transición de nivel BAJO a nivel ALTO, cuando el sistema pasa a los estados primero (00) o tercero (11). La salida para disparar el circuito temporizador de 4 s. (temporizador corto) da lugar a una transición de nivel BAJO a nivel ALTO cuando el sistema pasa a los estados segundo (01) o cuarto (10). Las salidas del circuito de disparo se muestran en la Tabla 6.11 y en forma de ecuación son:

$$\textit{Temporizador largo} = SO_1 + SO_3$$

$$\textit{Temporizador corto} = SO_2 + SO_4$$

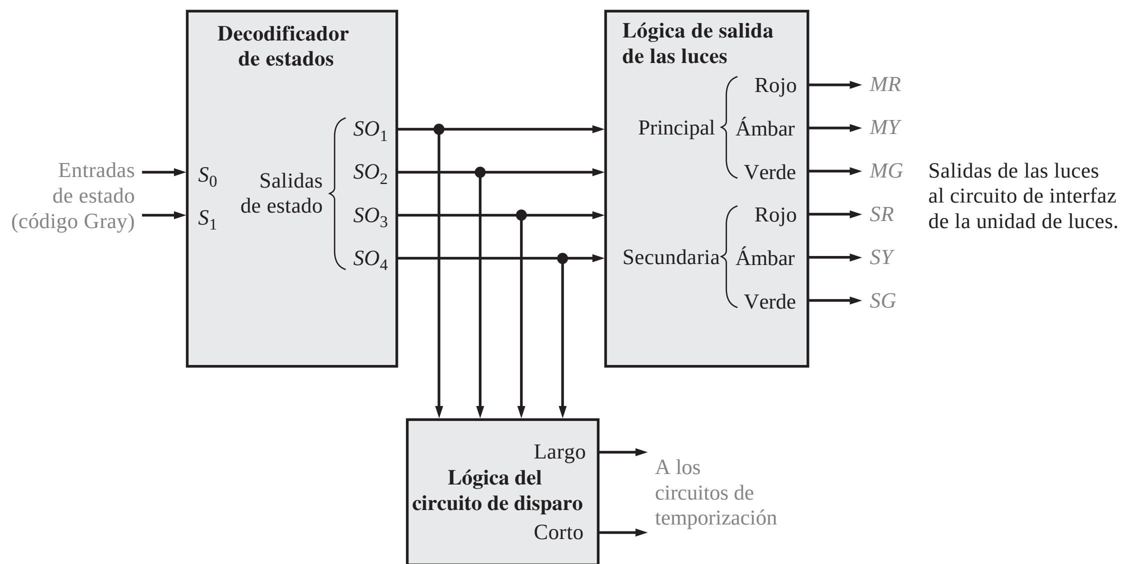


FIGURA 6.69 Diagrama de bloques para la lógica combinacional.

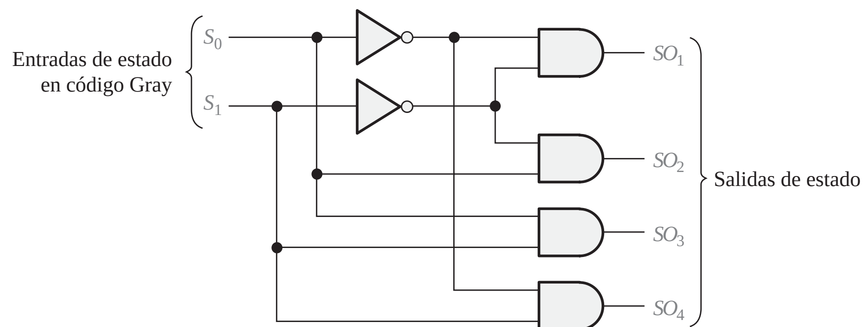


FIGURA 6.70 Lógica del decodificador de estados.

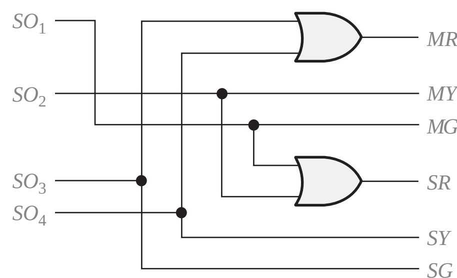


FIGURA 6.71 Lógica de salida de las luces.

Entradas de estado		Salidas de estado				Salidas de las luces						Salidas del circuito de disparo	
S_1	S_0	SO_1	SO_2	SO_3	SO_4	*MR*	*MY*	*MG*	*SR*	*SY*	*SG*	*LARGO*	*CORTO*
0	0	1	0	0	0	0	0	1	1	0	0	1	0
0	1	0	1	0	0	0	1	0	1	0	0	0	1
1	1	0	0	1	0	1	0	0	0	0	1	1	0
1	0	0	0	0	1	1	0	0	0	1	0	0	1

Las salidas de estado y las salidas de las luces son activas a nivel ALTO. *MR* corresponde al semáforo de la calle principal (*M)* cuando está en rojo (*R*), *SG* corresponde al semáforo de la calle secundaria (*S*) cuando está en verde (*G*).

TABLA 6.11 Tabla de verdad de la lógica combinacional.

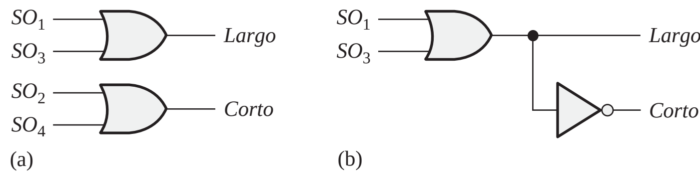


FIGURA 6.72 Lógica del circuito de disparo.

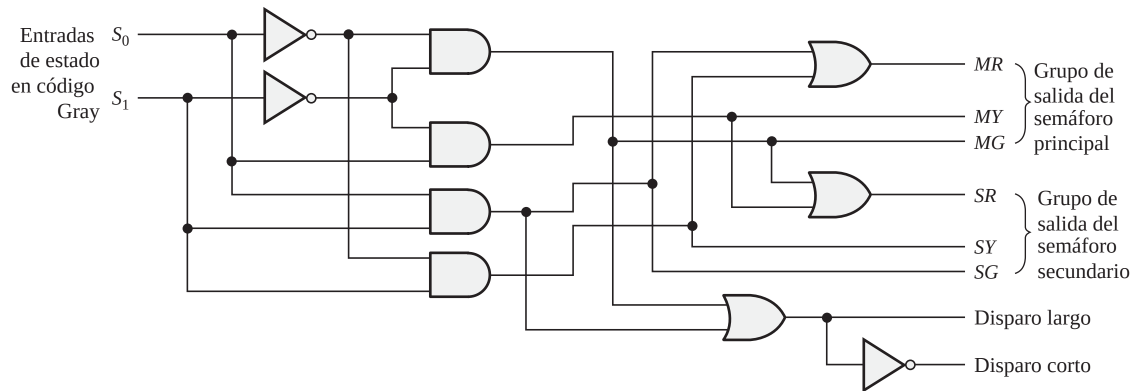


FIGURA 6.73 Lógica combinacional completa.

En la Figura 6.72(a) se muestra la lógica del circuito de disparo. La Tabla 6.11 también muestra que las salidas *LARGO* y *CORTO* son complementarias, por lo que la lógica puede implementarse también con una puerta OR y un inversor, como se ilustra en la parte (b).

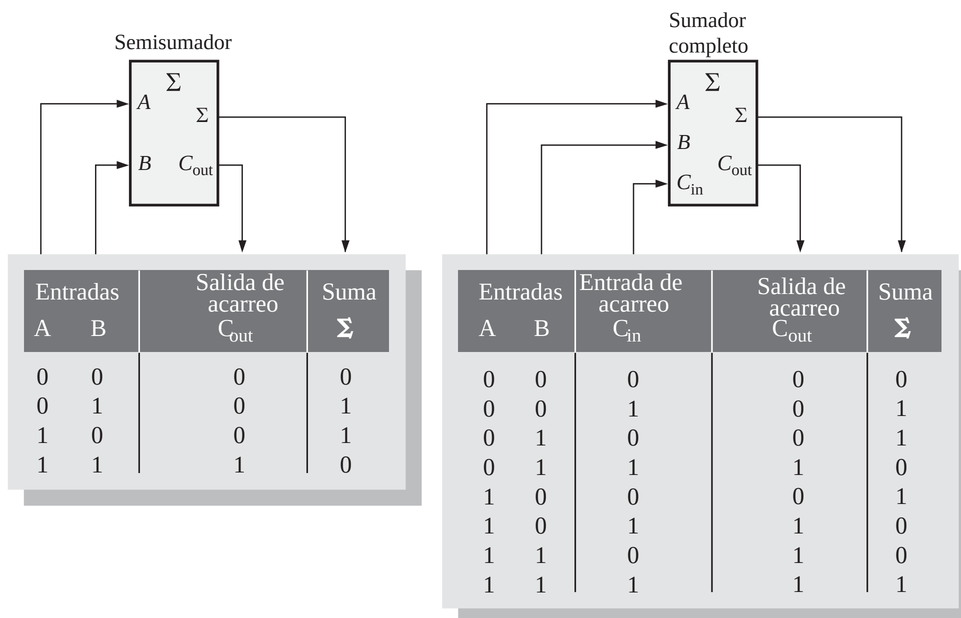
La Figura 6.73 muestra la lógica combinacional completa que combina el decodificador de estados, la lógica de salida de las luces y la lógica de disparo.

Prácticas de sistemas

- ***Actividad 1***. Aplicar las formas de onda del código Gray de 2 bits en las entradas S_0 y S_1 de la lógica combinacional y desarrollar todas las señales de salida.
- ***Actividad 2***. Demostrar cómo podría implementarse la lógica combinacional usando funciones 74XX.
- ***Actividad opcional***. Escribir un programa VHDL que describa la lógica combinacional.

RESUMEN

- El funcionamiento del semi-sumador y del sumador completo se muestran en la Figura 6.74.
- En la Figura 6.75 se muestran los símbolos lógicos y la numeración de pines de los CI empleados en este capítulo. La designación de pines puede variar dependiendo del fabricante.



Entradas A	Entradas B	Salida de acarreo C_{out}	Suma Σ
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Entradas A	Entradas B	Entrada de acarreo C_{in}	Salida de acarreo C_{out}	Suma Σ
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

FIGURA 6.74

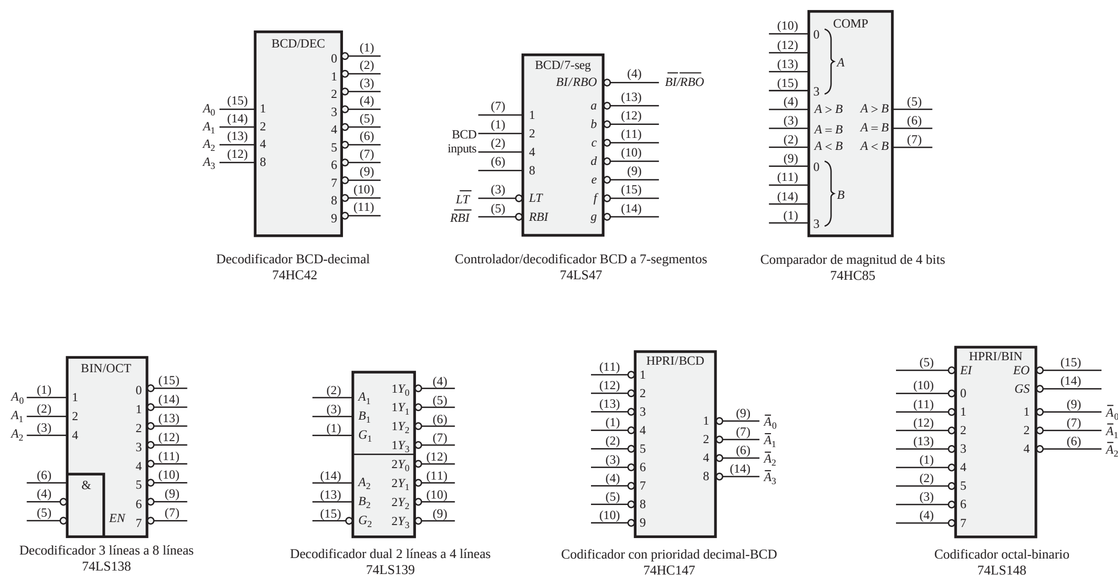
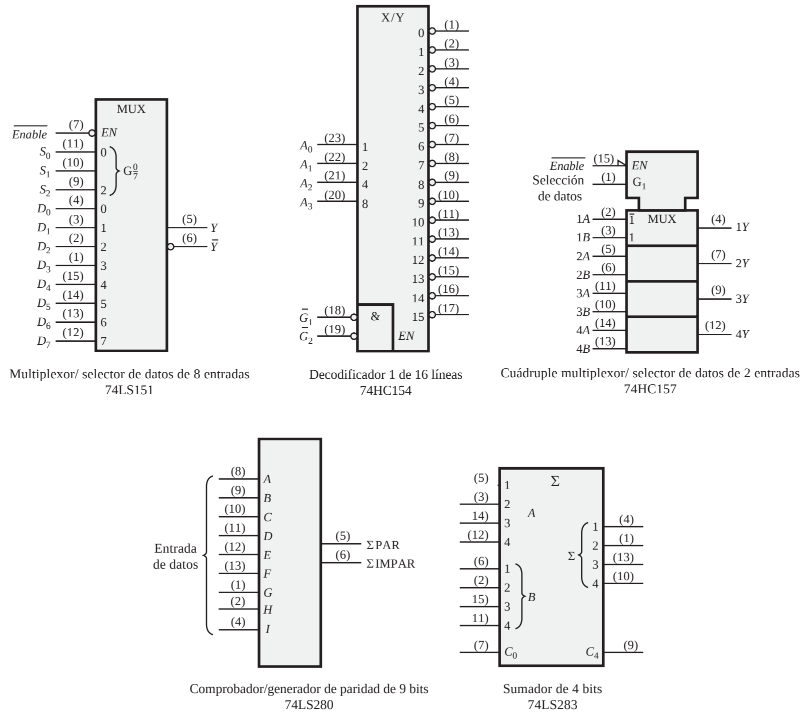


FIGURA 6.75 *(Continúa)*



Multiplexor/ selector de datos de 8 entradas 74LS151

Decodificador 1 de 16 líneas 74HC154

Cuádruple multiplexor/ selector de datos de 2 entradas 74HC157

Comprobador/generador de paridad de 9 bits 74LS280

Sumador de 4 bits 74LS283

FIGURA 6.75 *(Continuación)*

PALABRAS CLAVE

Las palabras clave y otros términos que se han resaltado en negrita se encuentran en el glosario final del libro.

Acarreo anticipado Método de suma binaria en que los acarreos de las etapas sumadoras precedentes se anticipan, eliminando de esta manera los retardos en la propagación de los acarreos.

Acarreo serie Método de suma binaria en el que el acarreo de salida de cada sumador se convierte en el acarreo de entrada del sumador siguiente.

Bit de paridad Bit que se añade a cada grupo de bits de información para hacer que el número de unos sea par o impar en dicho grupo de bits.

Codificador Circuito digital que convierte la información a un formato codificado.

Codificador con prioridad Codificador en el que sólo se codifica el dígito de entrada de valor más alto, ignorándose cualquier otra entrada activa.

Conexión en cascada Conectar la salida de un dispositivo a la entrada de un dispositivo similar, permitiendo a uno de los dispositivos excitar a otro, para aumentar la capacidad de operación.

Decodificador Circuito digital que convierte la información codificada en un formato más familiar o no codificado.

Demultiplexor (DEMUX) Circuito que conmuta los datos digitales desde una línea de entrada a varias líneas de salida según una secuencia temporal especificada.

Glitch Un pico de corriente o de tensión de corta duración, generalmente indeseado y que se produce de forma no intencionada.

Multiplexor (MUX) Circuito que conmuta los datos digitales de distintas líneas de entrada a una única línea de salida según una secuencia temporal especificada.

Semisumador Circuito digital que suma dos bits y genera una suma y un acarreo de salida. No puede manipular acarreos de entrada.

Sumador completo Circuito digital que suma dos bits y un acarreo de entrada para producir una suma y un acarreo de salida.

AUTOTEST

Las respuestas se encuentran al final del capítulo.

1. Un semi-sumador se caracteriza por tener:
(a) dos entradas y dos salidas **(b)** tres entradas y dos salidas.
(c) dos entradas y tres salidas **(d)** dos entradas y una salida.

2. Un sumador completo se caracteriza por tener:
(a) dos entradas y dos salidas **(b)** tres entradas y dos salidas.
(c) dos entradas y tres salidas **(d)** dos entradas y una salida.

3. Las entradas de un sumador completo son $A = 1$, $B = 1$, $C_{in} = 0$. Sus salidas son:
(a) $\Sigma = 1$, $C_{out} = 1$ **(b)** $\Sigma = 1$, $C_{out} = 0$
(c) $\Sigma = 0$, $C_{out} = 1$ **(d)** $\Sigma = 0$, $C_{out} = 0$

4. Un sumador en paralelo de 4 bits puede sumar:
(a) dos números binarios de 4 bits **(b)** dos números binarios de 2 bits
(c) cuatro bits a la vez **(d)** una secuencia de cuatro bits

5. Para ampliar un sumador en paralelo de 4 bits a un sumador en paralelo de 8 bits, hay que:
(a) usar cuatro sumadores de 4 bits sin interconexiones
(b) usar dos sumadores de 4 bits y conectar las salidas de la suma de uno de ellos a las entradas de datos del otro.
(c) usar ocho sumadores de 4 bits sin interconexiones
(d) usar dos sumadores de 4 bits y conectar la salida de acarreo de uno de ellos a la entrada de acarreo del otro.

6. Si un comparador de magnitud 74HC85 tiene $A = 1011$ y $B = 1001$ en su entrada, las salidas son:
(a) $A > B = 0, A < B = 1, A = B = 0$ **(b)** $A > B = 1, A < B = 0, A = B = 0$
(c) $A > B = 1, A < B = 1, A = B = 0$ **(d)** $A > B = 0, A < B = 0, A = B = 1$

7. Si un decodificador de 4 líneas a 16 líneas con salidas activas a nivel BAJO persenta un nivel BAJO en la salida decimal 12, ¿cuáles son las entradas?
(a) $A_3A_2A_1A_0 = 1010$ **(b)** $A_3A_2A_1A_0 = 1110$
(c) $A_3A_2A_1A_0 = 1100$ **(d)** $A_3A_2A_1A_0 = 0100$

8. Un decodificador BCD a 7 segmentos tiene 0100 en sus entradas. Las salida activas serán:
(a) *a, c, f, g* **(b)** *b, c, f, g* **(c)** *b, c, e, f* **(d)** *b, d, e, g*

9. Si un codificador con prioridad octal-binario tiene en sus entradas 0, 2, 5 y 6 en un nivel activo, la salida binaria activa a nivel ALTO será:
(a) 110 **(b)** 010 **(c)** 101 **(d)** 000

10. En general, un multiplexor tiene
(a) una entrada de datos, varias salidas de datos y entradas de selección.
(b) una entrada de datos, una salida de datos y una entrada de selección.
(c) varias entradas de datos, varias salidas de datos y entradas de selección.
(d) varias entradas de datos, una salida de datos y entradas de selección.

11. Básicamente, los selectores de datos son lo mismo que:
    (a) decodificadores **(b)** demultiplexores
    (c) multiplexores **(d)** codificadores
12. ¿Cuáles de los códigos siguientes tienen paridad par?
    (a) 10011000 **(b)** 01111000 **(c)** 11111111
    (d) 11010101 **(e)** todos **(f)** las respuestas (b) y (c)

PROBLEMAS

Las respuestas a los problemas impares se encuentran al final del libro.

SECCIÓN 6.1 Sumadores básicos

1. Para el sumador completo de la Figura 6.4, determinar el estado lógico (1 o 0) a la salida de cada puerta para las siguientes entradas:
   (a) $A = 1, B = 1, C_{in} = 1$ **(b)** $A = 0, B = 1, C_{in} = 1$ **(c)** $A = 0, B = 1, C_{in} = 0$
2. ¿Cuáles serían las entradas que producirían en un sumador completo las siguientes salidas?
   (a) $\Sigma = 0, C_{out} = 0$ **(b)** $\Sigma = 1, C_{out} = 0$ **(c)** $\Sigma = 1, C_{out} = 1$ **(d)** $\Sigma = 0, C_{out} = 1$
3. Determinar las salidas de un sumador completo para cada una de las siguientes entradas:
   (a) $A = 1, B = 0, C_{in} = 0$ **(b)** $A = 0, B = 0, C_{in} = 1$
   (a) $A = 0, B = 1, C_{in} = 1$ **(d)** $A = 1, B = 1, C_{in} = 1$

SECCIÓN 6.2 Sumadores binarios en paralelo

4. Para el sumador en paralelo de la Figura 6.76, determinar la suma completa mediante el análisis del funcionamiento lógico del circuito. Comprobar el resultado sumando manualmente los dos números de entrada.

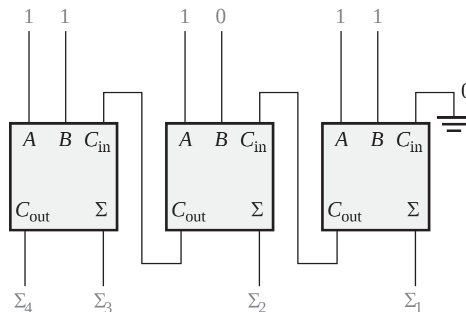


FIGURA 6.76

5. Repetir el Problema 4 para el circuito y las condiciones de entrada de la Figura 6.77.

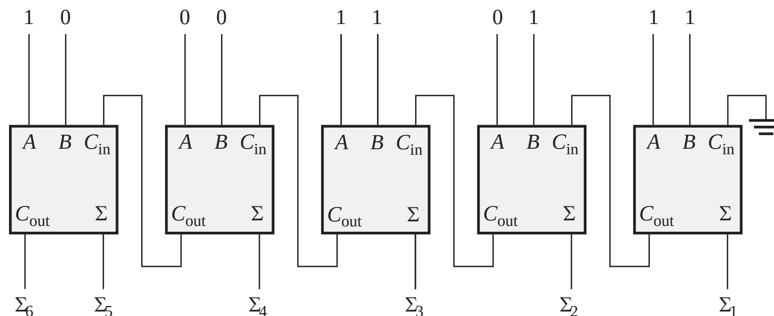


FIGURA 6.77

6. Las formas de onda de entrada de la Figura 6.78 se aplican a un sumador de 2 bits. Determinar, mediante un diagrama de tiempo, las señales correspondientes a la suma y a la salida de acarreo, en función de las entradas.

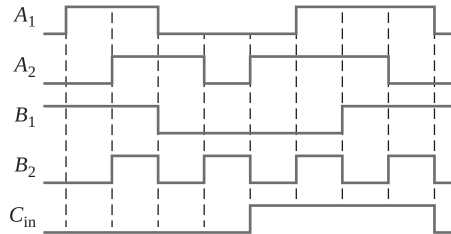


FIGURA 6.78

7. Las siguientes secuencias de bits (el bit de la derecha es el primero) se aplican a las entradas de un sumador en paralelo de 4 bits. Determinar la secuencia de bits resultante en cada salida.

A_1	1001
A_2	1110
A_3	0000
A_4	1011
B_1	1111
B_2	1100
B_3	1010
B_4	0010

8. En las pruebas de un sumador completo de 4 bits 74LS83, se observan los siguientes niveles de tensión en sus pines: 1-ALTO, 2-ALTO, 3-ALTO, 4-ALTO, 5-BAJO, 6-BAJO, 7-BAJO, 9-ALTO, 10-BAJO, 11-ALTO, 12-BAJO, 13-ALTO, 14-ALTO y 15-ALTO. Determinar si el circuito integrado funciona correctamente.

SECCIÓN 6.3 Sumadores de acarreo serie y acarreo anticipado

9. Cada uno de los ocho sumadores completos de un sumador de 8 bits con acarreo anticipado presenta los siguientes retardos de propagación:

A a Σ y C_{out}:	40 ns
B a Σ y C_{out}:	40 ns
C_{in}:	35 ns
C_{in} a C_{out}:	25 ns

Determinar el tiempo total máximo necesario para sumar dos números de 8 bits.

10. Indicar qué circuitería adicional es necesaria para convertir en sumador de 4 bits de acarreo anticipado de la Figura 6.18 en un sumador de 5 bits.

SECCIÓN 6.4 Comparadores

11. Se aplican las formas de onda mostradas en la Figura 6.79 a las entradas del comparador. Determinar la señal de salida (A=B).

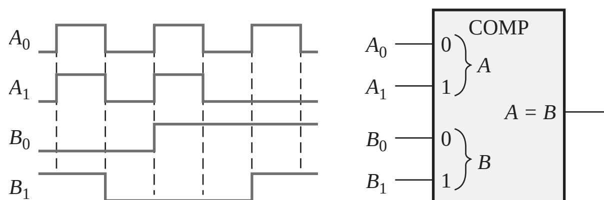


FIGURA 6.79

12. Para el comparador de 4 bits de la Figura 6.80, dibujar cada forma de onda de salida para las entradas que se muestran. Las salidas son activas a nivel ALTO.

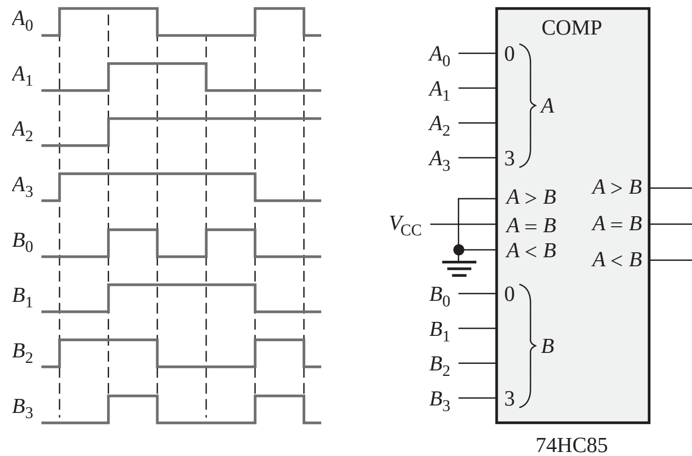


FIGURA 6.80

13. Para los siguientes grupos de números binarios, determinar los estados de salida para el comparador de la Figura 6.22.

(a) $A_3A_2A_1A_0 = 1100$; $B_3B_2B_1B_0 = 1001$

(b) $A_3A_2A_1A_0 = 1000$; $B_3B_2B_1B_0 = 1011$

(c) $A_3A_2A_1A_0 = 0100$; $B_3B_2B_1B_0 = 0100$

SECCIÓN 6.5 Decodificadores

14. Cuando en la salida de cada puerta de decodificación de la Figura 6.81 hay un nivel ALTO, ¿cuál es el código binario que aparece en sus entradas? El bit más significativo (MSB) es A_3.

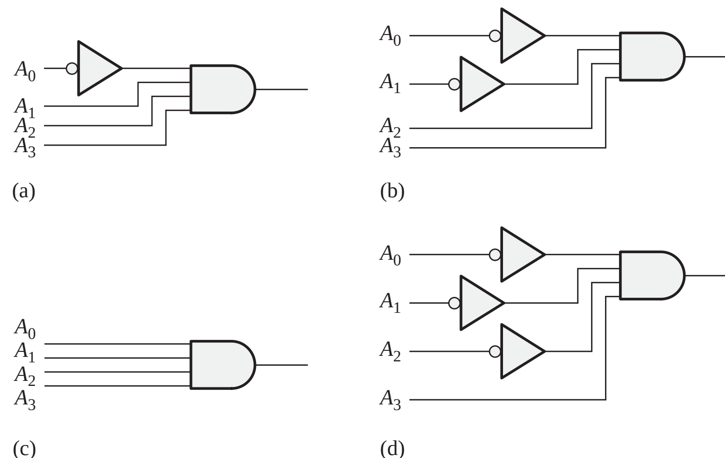


FIGURA 6.81

15. ¿Cuál es la lógica de decodificación para cada uno de los siguientes códigos, si se requiere una salida activa a nivel ALTO (1)?

(a) 1101 **(b)** 1000 **(c)** 11011 **(d)** 11100

(e) 101010 **(f)** 111110 **(g)** 000101 **(h)** 1110110

16. Resolver el Problema 13, suponiendo que se requiere una salida activa a nivel BAJO (0).

17. Se desea detectar únicamente la presencia de los códigos 1010, 1100, 0001 y 1011. Para indicar la presencia de dichos códigos se requiere una salida activa a nivel ALTO. Desarrollar la lógica de decodificación mínima necesaria que tenga una única salida que indique cuándo cualquiera de estos códigos se encuentra en las entradas. Para cualquier otro código, la salida ha de ser un nivel BAJO.

18. Si se aplican las formas de onda de entrada a la lógica de decodificación de la Figura 6.82, dibujar las formas de onda de salida en función de dichas entradas.

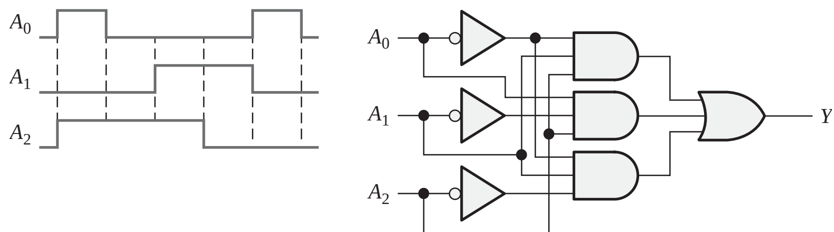


FIGURA 6.82

19. Se aplican secuencialmente números BCD al decodificador BCD-decimal de la Figura 6.83. Dibujar un diagrama de tiempos que muestre cada salida en relación con el resto de las señales de salida y con las de entrada.

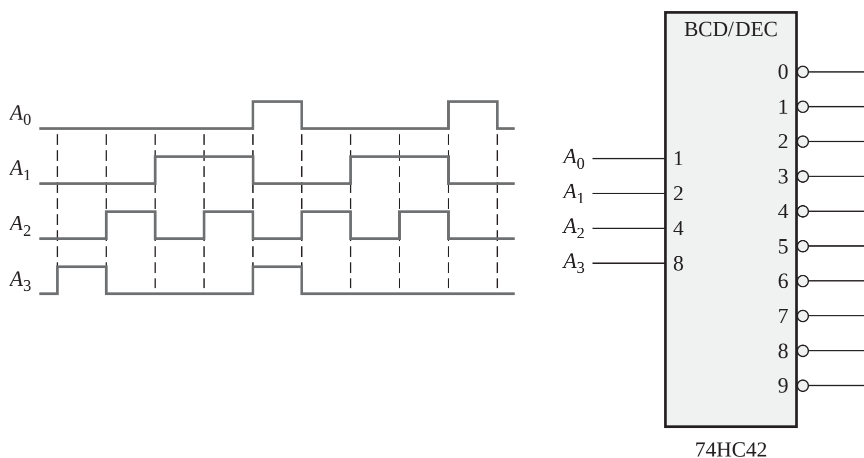


FIGURA 6.83

20. Un decodificador/excitador de 7-segmentos controla el display de la Figura 6.84. Si se aplican las formas de onda de entrada que se muestran, determinar la secuencia de dígitos que aparece en el display.

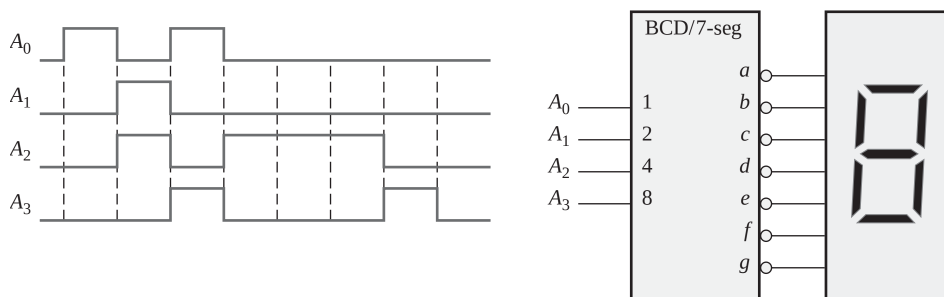


FIGURA 6.84

SECCIÓN 6.6 Codificadores

21. Suponer que el codificador lógico decimal-BCD de la Figura 6.38 tiene las entradas 3 y 9 a nivel ALTO. ¿Cuál es el código de salida? ¿Es éste un código BCD (8421) válido?

22. Un decodificador 74HC147 tiene niveles BAJOS de tensión en sus pines 2, 5 y 12. ¿Qué código BCD aparece en las salidas si todas las demás entradas están a nivel ALTO?

SECCIÓN 6.7 Convertidores de código

23. Convertir a BCD los siguientes números decimales y luego a binario.

(a) 2 **(b)** 8 **(c)** 13

(d) 26 **(e)** 33

24. Explicar la lógica requerida para convertir a código Gray un número binario de 10 bits, y utilizar esta lógica para convertir los siguientes números binarios:

(a) 1010101010 **(b)** 1111100000

(c) 0000001110 **(d)** 1111111111

25. Explicar la lógica requerida para convertir a binario un código Gray de 10 bits y utilizar esta lógica para convertir a binario los siguientes códigos Gray:

(a) 1010000000 **(b)** 0011001100

(c) 1111000111 **(d)** 0000000001

SECCIÓN 6.8 Multiplexores (selectores de datos)

26. En el demultiplexor de la Figura 6.85, determinar la salida para los siguientes estados de entrada: $D_0 = 0$, $D_1 = 1$, $D_2 = 1$, $D_3 = 0$, $S_0 = 1$, $S_1 = 0$

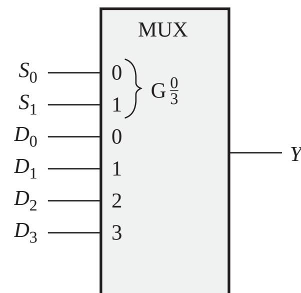


FIGURA 6.85

27. Si las entradas de selección de datos del multiplexor de la Figura 6.85 se secuencian tal y como se muestra en las formas de onda de la Figura 6.86, determinar la forma de onda de salida para los datos de entrada del Problema 26.

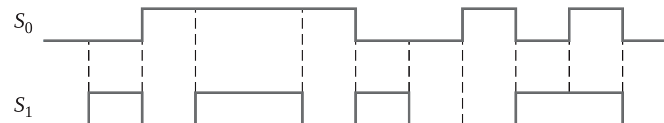


FIGURA 6.86

28. Las formas de onda mostradas en la Figura 6.87 se aplican a las entradas de un multiplexor de ocho entradas 74LS151. Dibujar la señal de salida *Y*.

SECCIÓN 6.9 Demultiplexores

29. Desarrollar el diagrama de tiempos completo (entradas y salidas) de un 74HC154 utilizado en una aplicación de demultiplexación en el que las entradas son las siguientes: las entradas de selección de datos toman, de forma repetitiva y secuencialmente, los valores generados por un

contador binario que comienza en 0000, y la entrada de datos es una cadena de datos serie, en BCD, que representan al número decimal 2468. El dígito menos significativo (8) es el primero de la secuencia, con el bit menos significativo en primer lugar, y deberá aparecer en los cuatro primeros bits de la salida.

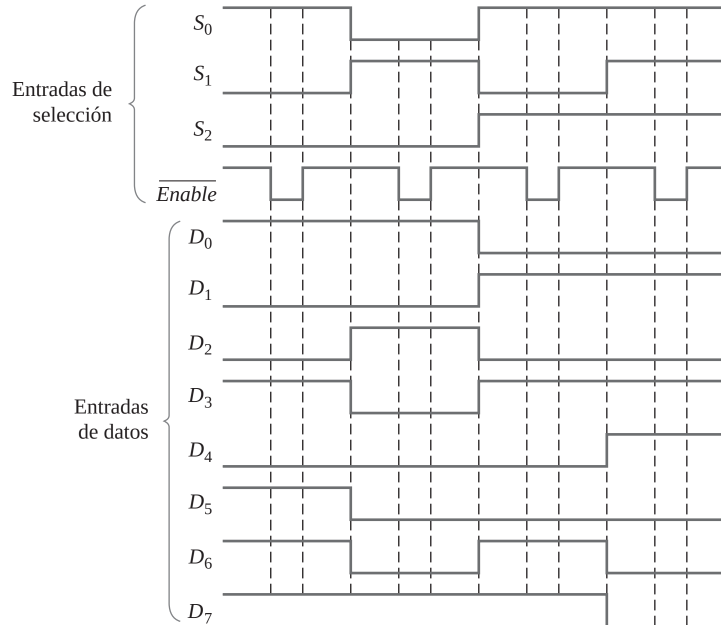


FIGURA 6.87

SECCIÓN 6.10 Generadores / Comprobadores de paridad

30. Se aplican las formas de onda de la Figura 6.88 al circuito de paridad de 4 bits. Determinar las señales de salida en función de las entradas. ¿Durante cuántos periodos de bit ocurre la paridad par y cómo se indica? El diagrama de tiempos incluye ocho periodos de bit.

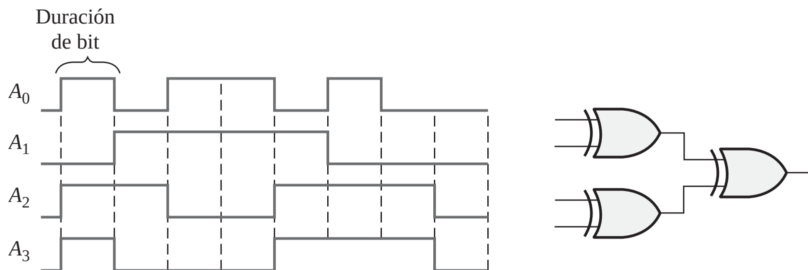


FIGURA 6.88

31. Determinar las salidas ΣImpar y ΣPar de un generador/comprobador de paridad 74LS280 de 9 bits, para las entradas de la Figura 6.89. Utilice la tabla de verdad de la Figura 6.59.

SECCIÓN 6.11 Localización de averías

32. El sumador completo de la Figura 6.90 se prueba bajo todas las condiciones de entrada posibles, con las señales de entrada indicadas. A partir de la observación de las señales Σ y C_{out}, ¿funciona correctamente? Si la respuesta es no, ¿cuál es la causa más probable de fallo?

33. Enumerar los posibles fallos de cada codificador/display de la Figura 6.91.

34. Desarrollar un procedimiento de pruebas sistemático para verificar el funcionamiento completo del codificador de teclado de la Figura 6.42.

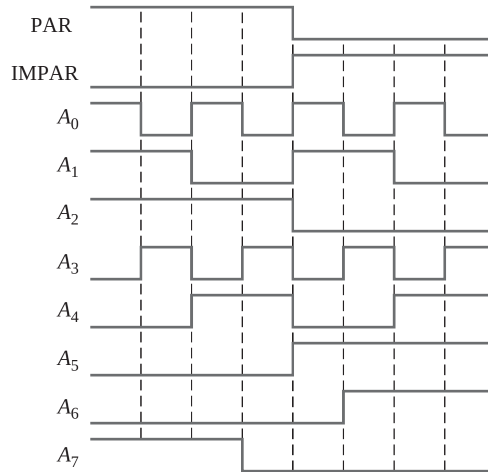


FIGURA 6.89

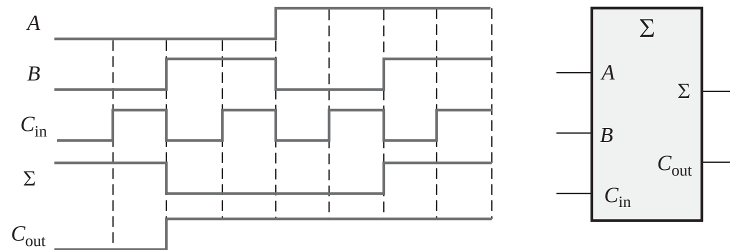


FIGURA 6.90

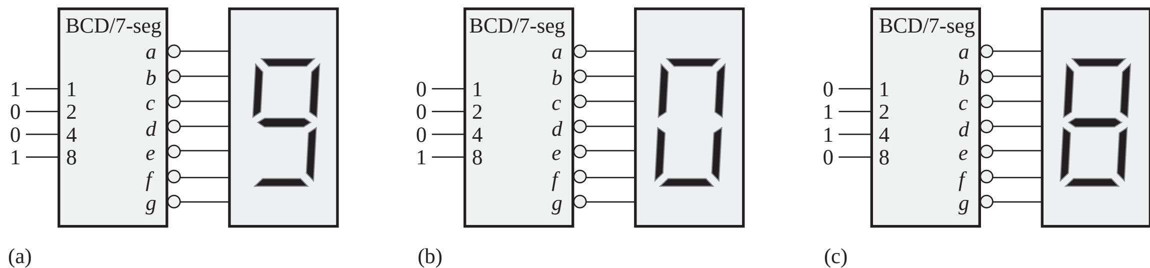


FIGURA 6.91

35. Hay que probar el convertidor BCD-binario formado por cuatro sumadores que se muestra en la Figura 6.92. En primer lugar, hay que verificar que el circuito convierte de BCD a binario. El procedimiento de prueba requiere la aplicación secuencial de números BCD, comenzando por 0_{10}, para comprobar que la salida binaria es la correcta. ¿Qué síntoma o síntomas aparecerían en las salidas binarias si ocurrieran cada uno de los siguientes fallos? ¿Cuál es el número BCD para el que se detecta por *primera vez* cada error?

(a) La entrada A_1 está en circuito abierto (sumador superior).

(b) C_{out} está en circuito abierto (sumador superior).

(c) La salida Σ_4 está cortocircuitada a masa (sumador superior).

(d) La salida 32 está cortocircuitada a masa (sumador inferior).

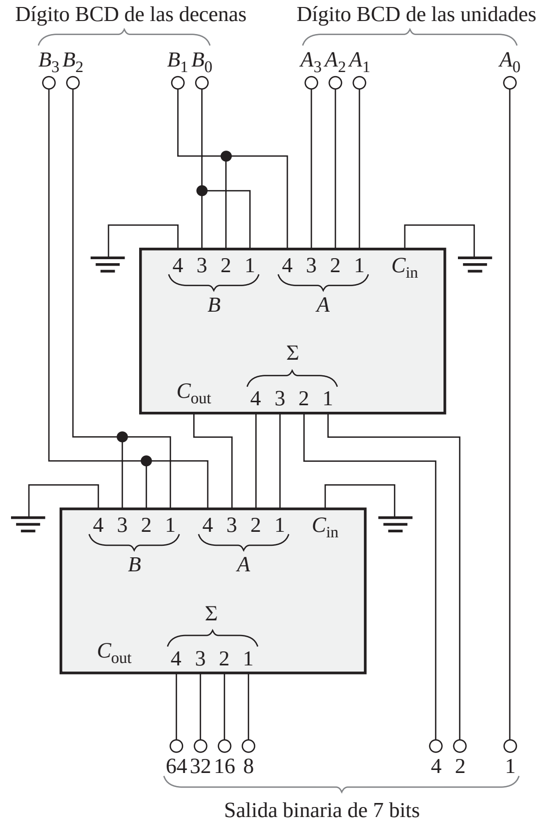


FIGURA 6.92

36. En el display con multiplexación de la Figura 6.52, determinar la causa (o las causas) más probable para cada uno de los siguientes síntomas:

(a) El display del dígito *B* (el más significativo) no se enciende.

(b) Ninguno de los displays de 7-segmentos se enciende.

(c) El segmento *f* de ambos displays aparece encendido siempre.

(d) Hay un parpadeo visible en los displays.

37. Desarrollar un procedimiento sistemático para probar exhaustivamente el CI selector de datos 74LS151.

38. Durante las pruebas del sistema de transmisión de datos de la Figura 6.60, se aplica un código a las entradas D_0 a D_6 que contiene un número impar de 1s. Se introduce deliberadamente un único bit erróneo en la línea de transmisión serie entre el multiplexor y el demultiplexor, pero el sistema no detecta el error (salida de error = 0). Tras algún tiempo de investigación, se verifican las entradas con el comprobador de paridad par y se encuentra que en D_0 a D_6 hay un número par de 1s, como se esperaba, y también se comprueba que el bit de paridad D_7 es 1. ¿Cuáles son las posibles razones de que el sistema no indique el error?

39. Describir de forma general cómo probaríamos el sistema de transmisión de datos de la Figura 6.60 y especificar un método de introducción de errores de paridad.



Aplicación a los sistemas digitales

40. El bloque de la lógica de salida del semáforo se implementa en el sistema usando lógica de función fija mediante un 74LS08 con puertas AND operando como puertas negativa-NOR.

Utilizar un 74LS00 (puertas NAND cuádruples) y cualquier otro dispositivo que sea necesario para generar salidas activas a nivel BAJO para las entradas dadas.

41. Implementar la lógica de salida del semáforo con el 74LS00 si se necesitan salidas activas a nivel BAJO.



Problemas especiales de diseño

42. Modificar el diseño del sistema de multiplexación del display de 7-segmentos de la Figura 6.52 para permitir visualizar dos dígitos adicionales.

43. Utilizando la Tabla 6.2, escribir las expresiones de suma de productos para Σ y C_{out} de un sumador completo. Utilizar un mapa de Karnaugh para minimizar las expresiones y luego implementarlas empleando inversores y lógica AND-OR. Indicar cómo se puede reemplazar la lógica AND-OR con selectores de datos 74LS151.

44. Implementar la función lógica especificada en la Tabla 6.12 utilizando un selector de datos 74LS151.

Entradas				Salida
A_3	A_2	A_1	A_0	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	1

TABLA 6.12

45. Utilizando dos de los módulos sumadores de 6 posiciones de la Figura 6.14, diseñar un sistema de votación de 12 posiciones.

46. El bloque sumador del sistema de control y recuento de pastillas de la Figura 6.93 realiza la suma del número binario de 8 bits del contador y del número binario de 16 bits del registro B. El resultado de la suma se almacena en el registro B. Utilizar circuitos 74LS283 para implementar esta función y dibujar un diagrama lógico completo que incluya la numeración de los pines. Revise el funcionamiento del sistema en el Capítulo 1.

47. Utilizar circuitos 74HC85 para implementar el bloque comparador del sistema de control y recuento de pastillas de la Figura 6.93 y dibujar un diagrama lógico completo que incluya la numeración de pines. El comparador compara el número binario de 8 bits (en realidad sólo se requieren siete bits) del convertidor BCD-binario con el número binario de 8 bits del contador.

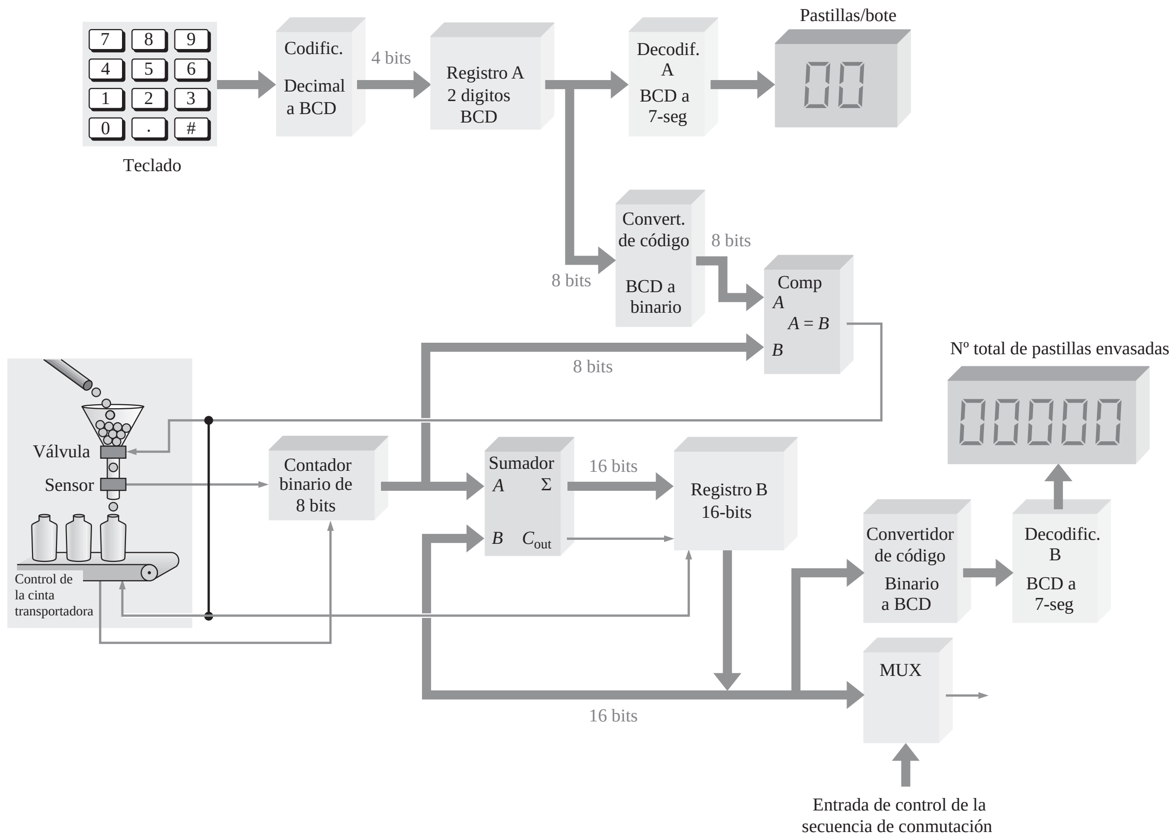


FIGURA 6.93

48. Se utilizan dos decodificadores BCD-7-segmentos en el sistema de control y recuento de la Figura 6.93. Uno de ellos se requiere para controlar el display de dos dígitos *pastillas/bote*, y el otro para controlar el display de 5 dígitos *número total de pastillas envasadas*. Utilizar circuitos 74LS47 para implementar cada decodificador y dibujar un diagrama lógico completo que incluya la numeración de pines.

49. El codificador que se muestra en el diagrama de bloques de la Figura 6.93 codifica cada pulsación de una tecla decimal y la convierte en BCD. Utilizar un 74HC147 para implementar esta función y dibujar un diagrama lógico completo que incluya la numeración de pines.

50. El sistema de la Figura 6.93 requiere dos convertidores de código. El convertidor BCD-binario convierte los dos números BCD de dos dígitos del registro *A* en un código binario de 8 bits (en realidad sólo se necesitan 7 bits dado que el bit más significativo siempre es 0). Utilizar los convertidores de código circuito integrado apropiados para implementar la función del convertidor BCD-binario, y dibujar un diagrama lógico completo que incluya la numeración de pines.

RESPUESTAS

REVISIONES DE CADA SECCIÓN

SECCIÓN 6.1 Sumadores básicos

1. **(a)** $\Sigma = 1$, $C_{out} = 0$ **(b)** $\Sigma = 0$, $C_{out} = 0$

(c) $\Sigma = 1$, $C_{out} = 0$ **(d)** $\Sigma = 0$, $C_{out} = 1$

2. $\Sigma = 1$, $C_{out} = 1$

SECCIÓN 6.2 Sumadores binarios en paralelo

1. $C_{out}\Sigma_4\Sigma_3\Sigma_2\Sigma_1 = 11001$

2. Se requieren tres 74LS283 para sumar dos números de 10 bits.

SECCIÓN 6.3 Sumadores de acarreo serie y acarreo anticipado

1. $C_g = 0$, $C_p = 1$

2. $C_{out} = 1$

SECCIÓN 6.4 Comparadores

1. $A > B = 1$, $A < B = 0$, $A = B = 0$ cuando $A = 1011$ y $B = 1010$

2. Comparador de la derecha: pin 7: $A < B = 1$; pin 6: $A = B = 0$; pin 5: $A > B = 0$
Comparador de la izquierda: pin 7: $A < B = 0$; pin 6: $A = B = 0$; pin 5: $A > B = 1$

SECCIÓN 6.5 Decodificadores

1. La salida 5 está activa cuando en las entradas se aplica 101.

2. Se utilizan cuatro 74HC154 para decodificar un número binario de 6 bits.

3. La salida activa a nivel BAJO controla el display de diodos LED en cátodo común.

SECCIÓN 6.6 Codificadores

1. (a) $A_0 = 1$, $A_1 = 1$, $A_2 = 0$, $A_3 = 1$

(b) No, no es un código BCD válido.

(c) Sólo puede estar activada una entrada para obtener una salida válida.

2. **(a)** $\overline{A}_3 = 0$, $\overline{A}_2 = 1$, $\overline{A}_1 = 1$, $\overline{A}_0 = 1$

(b) La salida es 0111, que es el complemento de 1000 (8).

SECCIÓN 6.7 Convertidores de código

1. 10000101 (BCD) = 1010101_2

2. Un convertidor binario-código Gray de ocho bits está formado por siete puertas OR-exclusiva en una disposición como la de la Figura 6.43.

SECCIÓN 6.8 Multiplexores (selectores de datos)

1. La salida es 0.

2. **(a)** 74LS157: Selector de datos cuádruple de 2 entradas.

(b) 74LS151: Selector de datos de 8 entradas.

3. La salida de datos alterna entre un nivel BAJO y un nivel ALTO a medida que las entradas de selección de datos cambian, secuencialmente, entre los distintos estados binarios.

4. **(a)** El 74HC157A multiplexa los dos códigos BCD al decodificador de 7-segmentos.

(b) El 74LS47 decodifica el código BCD para excitar el display.

(c) El 74LS139 activa los displays de 7-segmentos alternativamente.

SECCIÓN 6.9 Demultiplexores

1. Se puede utilizar un decodificador como multiplexor, utilizando las líneas de entrada como entradas de selección de datos y una línea de activación como entrada de datos.

2. Las salidas están todas a nivel ALTO excepto D_{10}, que es un nivel BAJO.

SECCIÓN 6.10 Comprobadores/generadores de paridad

1. **(a)** Paridad par: 1110100
   (b) Paridad par: 001100011
2. **(a)** Paridad impar: 11010101
   (b) 11000001
3. **(a)** El código es correcto, cuatro 1s.
   (b) El código es erróneo, siete 1s.

SECCIÓN 6.11 Localización de averías

1. Un *glitch* es un pico de tensión de muy corta duración (generalmente indeseado).
2. Los *glitches* los originan los estados de transición.
3. Validar (*strobing*) consiste en la activación de un dispositivo durante un período de tiempo especificado, mientras el dispositivo no se encuentra en un estado de transición

PROBLEMAS RELACIONADOS

6.1 $\Sigma = 1$, $C_{out} = 1$

6.2 $\Sigma_1 = 0$, $\Sigma_2 = 0$, $\Sigma_3 = 1$, $\Sigma_4 = 1$

6.3 1011 + 1010 = 10101

6.4. *Véase* la Figura 6.94.

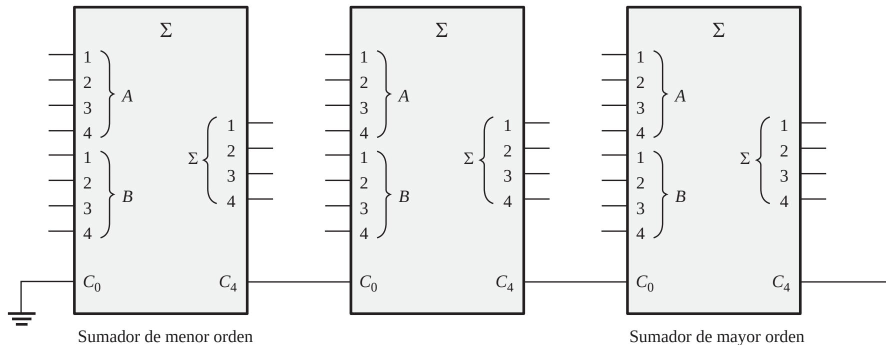


FIGURA 6.94

6.5. *Véase* la Figura 6.95.

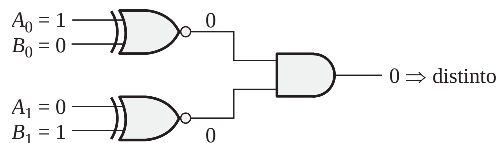


FIGURA 6.95

6.6 $A > B = 0$, $A = B = 0$, $A < B = 1$

6.7. *Véase* la Figura 6.96.

6.8. *Véase* la Figura 6.97.

6.9. Salida 22

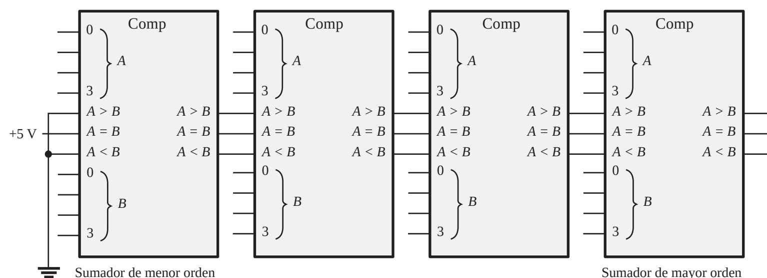


FIGURA 6.96

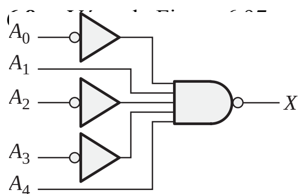


FIGURA 6.97

6.10. *Véase* la Figura 6.98.

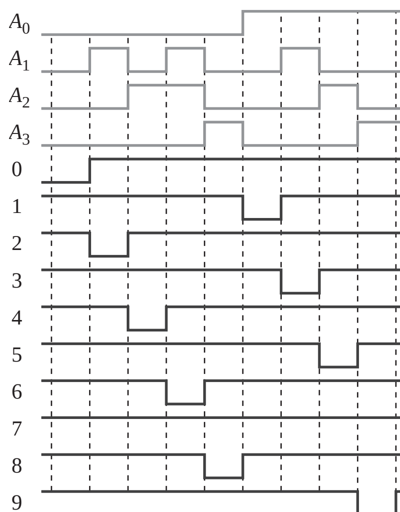


FIGURA 6.98

6.11. Todas las entradas a nivel BAJO: $\overline{A}_0 = 0, \overline{A}_1 = 1, \overline{A}_2 = 1, \overline{A}_3 = 0$

Todas las entradas a nivel ALTO: todas las salidas a nivel ALTO.

6.12 BCD 01000001

→	00000001	1
→	<u>00101000</u>	<u>40</u>
Binario	00101001	41

6.13 Siete puertas OR-exclusiva.

6.14. *Véase* la Figura 6.99.

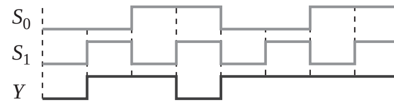


FIGURA 6.99

6.15. D_0: $S_3 = 0, S_2 = 0, S_1 = 0, S_0 = 0$
D_4: $S_3 = 0, S_2 = 1, S_1 = 0, S_0 = 0$
D_8: $S_3 = 1, S_2 = 0, S_1 = 0, S_0 = 0$
D_{13}: $S_3 = 1, S_2 = 1, S_1 = 0, S_0 = 1$

6.16. *Véase* la Figura 6.100.

6.17. *Véase* la Figura 6.101.

6.18. *Véase* la Figura 6.102.

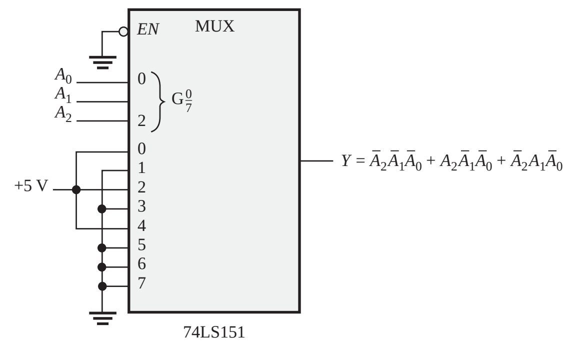


FIGURA 6.100

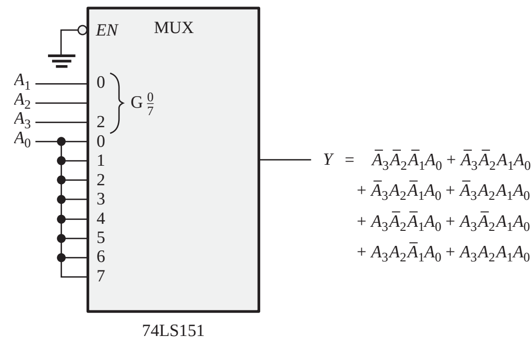


FIGURA 6.101

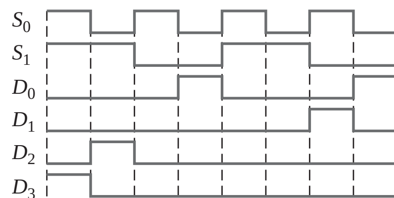


FIGURA 6.102

AUTOTEST

1. (a) **2.** (b) **3.** (c) **4.** (a) **5.** (d) **6.** (b) **7.** (c)
8. (b) **9.** (a) **10.** (d) **11.** (c) **12.** (f)

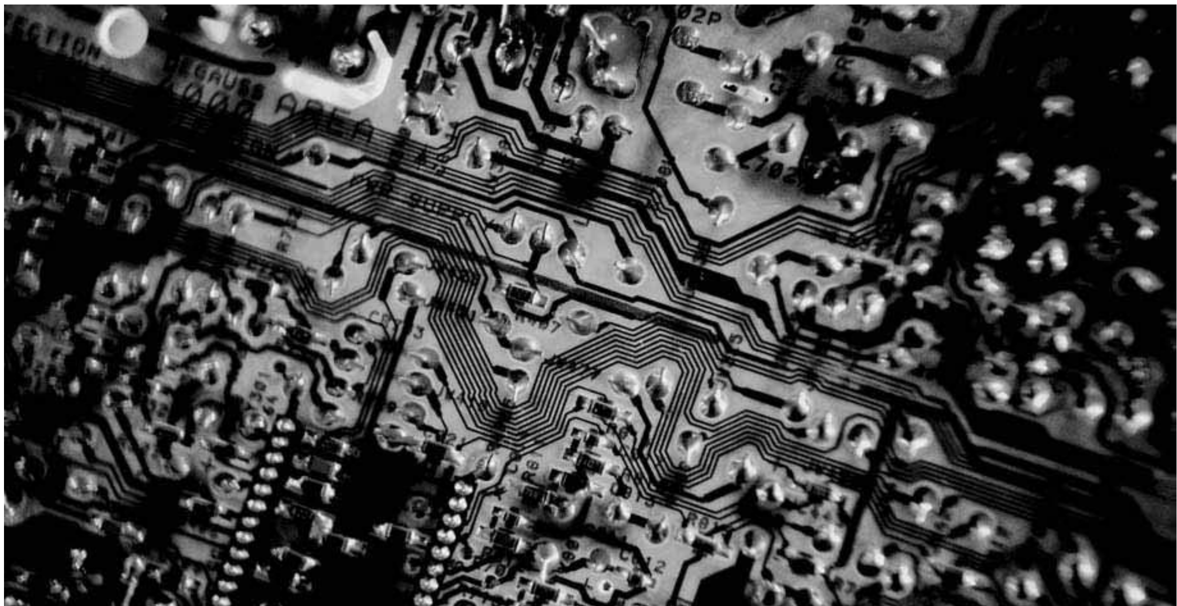
7

LATCHES, FLIP-FLOPS Y TEMPORIZADORES

CONTENIDO DEL CAPÍTULO

OBJETIVOS DEL CAPÍTULO

- Utilizar puertas lógicas para construir latches básicos.
- Explicar la diferencia entre un latch S-R y un latch D.
- Conocer las diferencias entre un latch y un flip-flop.
- Explicar en qué se diferencian los flip-flops S-R, D y J-K.
- Comprender el significado de: retardo de propagación, tiempo de establecimiento (*setup time*),



tiempo de mantenimiento (*hold time*), frecuencia máxima de funcionamiento, ancho mínimo del impulso de reloj y disipación de potencia en las aplicaciones de los flip-flops.

- Emplear flip-flops en aplicaciones sencillas.
- Explicar en qué se diferencian los monoestables redisparables y no redisparables.
- Conectar un temporizador 555 para operar como multivibrador aestable o como monoestable.
- Localizar las averías en circuitos básicos de flip-flops.

PALABRAS CLAVE

- Latch
- Biestable
- SET
- RESET
- Reloj
- Flip-flop disparado por flanco
- Síncrono
- Flip-flop D
- Flip-flop J-K
- Bascular
- Inicialización (*Preset*)
- Borrado (*Clear*)
- Tiempo de retardo de propagación
- Tiempo de establecimiento
- Tiempo de mantenimiento
- Disipación de potencia
- Monoestable
- Temporizador
- Aestable

INTRODUCCIÓN

En este capítulo se inicia el estudio de los fundamentos de la lógica secuencial. Se cubren los circuitos biestables, monoestables y los dispositivos lógicos aestables, denominados *multivibradores*. Los dispositivos biestables se dividen en dos categorías: flip-flops y latches. Los biestables poseen dos estados estables, denominados SET (activación) y RESET (desactivación), en los cuales se pueden mantener indefinidamente, lo que les hace muy útiles como dispositivos de almacenamiento. La diferencia básica entre latches y flip-flops es la manera en que cambian de un estado a otro. Los flip-flops son los bloques básicos de construcción de los contadores, registros y otros circuitos de control secuencial, y se emplean también en ciertos tipos de memorias. El multivibrador monoestable, normalmente denominado monoestable, tiene un único estado estable. Un monoestable genera un único impulso de anchura controlada cuando se activa o dispara. El multivibrador aestable no tiene ningún estado estable y se emplea principalmente como oscilador, es decir, como generador de señales automantenido. Los osciladores de impulsos se emplean como fuentes de señales de temporización en los sistemas digitales.

DISPOSITIVOS LÓGICOS DE FUNCIÓN FIJA

74XX74	74XX279	74XX122
555	74121	74XX75
74XX112		

APLICACIÓN A LOS SISTEMAS DIGITALES

La aplicación a los sistemas digitales continúa con el sistema de control de semáforos del Capítulo 6. Este capítulo se ocupa del circuito de temporización del sistema que genera la señal de reloj, del intervalo de temporización largo para las luces rojas y verdes y del intervalo de temporización corto para las luces ámbar. El reloj se emplea como señal básica de temporización, que hace que la lógica secuencial del sistema pase a través de sus estados. La lógica secuencial se desarrollará en el Capítulo 8.

7.1 LATCHES

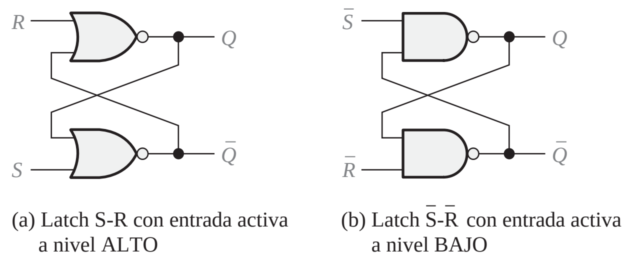
El ***latch*** (cerrojo) es un tipo de dispositivo de almacenamiento temporal de dos estados (biestable), que se suele agrupar en una categoría diferente a la de los flip-flops. Básicamente, los latches son similares a los flip-flops, ya que son también dispositivos de dos estados que pueden permanecer en cualquiera de sus dos estados gracias a su capacidad de realimentación, lo que consiste en conectar (realimentar) cada una de las salidas a la entrada opuesta. La diferencia principal entre ambos tipos de dispositivos está en el método empleado para cambiar de estado.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Explicar el funcionamiento de un latch S-R básico. ■ Explicar el funcionamiento de un latch S-R con entrada de habilitación. ■ Explicar el funcionamiento de un latch D con entrada de habilitación. ■ Implementar un latch S-R o D mediante puertas lógicas. ■ Describir los latches cuádruples 74LS279 y 74LS75.

El latch S-R (SET-RESET)

Un latch es un tipo de dispositivo lógico **biestable** o **multivibrador**. Un latch S-R (Set-Reset) con entrada activa a nivel ALTO se compone de dos puertas NOR acopladas, tal como se muestra en la Figura 7.1(a); un latch $\overline{S}-\overline{R}$ con entrada activa a nivel BAJO está formado por dos puertas NAND conectadas tal como se muestra en la Figura 7.1(b). Observe que la salida de cada puerta se conecta a la entrada de la puerta opuesta. Esto origina la **realimentación** (*feedback*) regenerativa característica de todos los latches y flip-flops.



(a) Latch S-R con entrada activa a nivel ALTO

(b) Latch $\overline{S}$-$\overline{R}$ con entrada activa a nivel BAJO

FIGURA 7.1 Dos versiones del latch S-R (SET-RESET).

Para explicar el funcionamiento del latch, vamos a utilizar el latch $\overline{S}-\overline{R}$ de puertas NAND de la Figura 7.1(b). Este latch lo hemos vuelto a dibujar en la Figura 7.2, utilizando puertas negativa-OR equivalentes, debido a que los niveles BAJOS de las líneas $\overline{S}$ *y* $\overline{R}$ son las entradas de activación.



NOTAS INFORMÁTICAS

Los dispositivos *latch* algunas veces se utilizan en sistemas informáticos para multiplexar datos sobre un bus. Por ejemplo, los datos que se introducen en una computadora desde una fuente externa tienen que compartir el bus de datos con datos procedentes de otras fuentes. Cuando el bus de datos no está disponible para la fuente externa, los datos existentes deben almacenarse temporalmente y para ello pueden colocarse *latches* entre la fuente externa y el bus de datos. Cuando el bus de datos no está disponible para la fuente externa, los *latches* deben desconectarse del bus utilizando un método conocido como tri-estado. Cuando el bus de datos vuelve a estar disponible, los datos externos pasan a través de los *latches*, lo que da lugar al uso del término *latch transparente*. El *latch* tipo D realiza esta función y cuando se activa, los datos que hay en su entrada aparecen en la salida del mismo modo que si se tratara de una conexión directa. Los datos de entrada se almacenan tan pronto como el *latch* se desactiva.

El latch de la Figura 7.2 tiene dos entradas, $\bar{S}$ y $\bar{R}$, y dos salidas Q y $\bar{Q}$. Asumimos que las dos entradas y la salida Q están a nivel ALTO. Dado que la salida Q se realimenta a una entrada de la puerta G_2 y que la entrada $\bar{R}$ está a nivel ALTO, la salida de G_2 tiene que ser un nivel BAJO. Esta salida a nivel BAJO está acoplada de nuevo a una entrada de la puerta G_1, asegurando así que su salida sea un nivel ALTO.

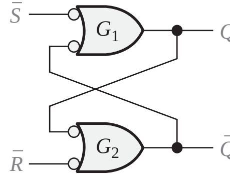


FIGURA 7.2 Equivalente con puertas negativa-OR para el latch $\bar{S}-\bar{R}$ de puertas NAND de la Figura 7.1(b).

▲ *Un latch puede permanecer en uno de dos estados, SET o RESET.*

Cuando la salida Q está a nivel ALTO, el latch se encuentra en estado ***SET*** y permanecerá indefinidamente en él hasta que se aplique un nivel BAJO a la entrada $\bar{R}$. Si tenemos un nivel BAJO en la entrada $\bar{R}$ y un nivel ALTO en $\bar{S}$, la salida de la puerta G_2 se pone forzosamente a nivel ALTO. Este nivel ALTO en la salida $\bar{Q}$ se realimenta a una de las entradas de G_1 y, dado que la entrada $\bar{S}$ está a nivel ALTO, la salida de G_1 se pone a nivel BAJO. Este nivel BAJO en la salida Q se realimenta a una de las entradas de G_2, asegurando que la salida $\bar{Q}$ permanezca a nivel ALTO incluso cuando se elimine el nivel BAJO de la entrada $\bar{R}$. Cuando la salida Q es un nivel BAJO, el latch se encuentra en estado ***RESET.*** Ahora el latch permanece indefinidamente en este estado hasta que se aplique un nivel BAJO en la entrada $\bar{S}$.

En operación normal, las salidas de un latch son siempre complementarias una de la otra:

Cuando Q está a nivel ALTO, $\bar{Q}$ está a nivel BAJO y cuando Q está a nivel BAJO, $\bar{Q}$ está a nivel ALTO.

▲ *SET indica que la salida Q está a nivel ALTO.*

Se produce una condición de funcionamiento no válida en un latch $\bar{S}-\bar{R}$ con entradas activas a nivel BAJO, cuando se aplican simultáneamente niveles bajos a las dos entradas, $\bar{S}$ y $\bar{R}$. Mientras que se mantengan las dos entradas a nivel BAJO, las dos salidas Q y $\bar{Q}$ deberían forzosamente estar a nivel ALTO, lo que viola la condición de complementariedad de las salidas. Además, si se eliminan simultáneamente los niveles BAJOS, las dos salidas van a tender al nivel BAJO y, dado que siempre va a existir un cierto retraso de propagación de la señal eléctrica a través de las puertas, una de las puertas dominará en la transición a nivel BAJO. Esto hará que la salida de la puerta más lenta permanezca a nivel ALTO. Cuando se produce esta situación, no se puede predecir el siguiente estado del latch.

▲ *RESET indica que la salida Q está a nivel BAJO.*

La Figura 7.3 ilustra el funcionamiento del latch $\bar{S}-\bar{R}$ con entradas activas a nivel BAJO, para cada una de las cuatro posibles combinaciones de los niveles de entrada. Las primeras tres combinaciones son válidas, no así la última. La Tabla 7.1 resume en forma de tabla de verdad el funcionamiento lógico. El funcionamiento del latch construido con puertas NOR con entradas activas a nivel ALTO de la Figura 7.1(a) es similar, pero requiere el uso de niveles lógicos opuestos.

Los símbolos lógicos para ambos tipos de latches, con entradas activas a nivel ALTO y a nivel BAJO, se muestran en la Figura 7.4.

El Ejemplo 7.1 ilustra cómo un latch $\bar{S}-\bar{R}$ con entradas activas a nivel BAJO responde a las condiciones de entrada. Los niveles BAJOS se aplican a las entradas siguiendo una determinada secuencia y se observa la señal de salida Q resultante. La condición $\bar{S}=0, \bar{R}=0$ no se contempla, ya que origina un modo de funcionamiento no válido del latch, lo que es un gran inconveniente en cualquier latch de tipo SET-RESET.

Pulso negativo

1 0 $\bar{S}$ G_1 Q 1 0 — Se produce una transición en la salida cuando $\bar{S}$ pasa a nivel BAJO, y permanece en el mismo estado después de que $\bar{S}$ pase a nivel ALTO.

$\bar{R}$ 1 (ALTO) G_2 $\bar{Q}$ 1 0

Estado inicial del latch: RESET ($Q = 0$).

1 0 $\bar{S}$ G_1 1 Q — No se producen transiciones porque el latch está ya en estado SET.

$\bar{R}$ 1 G_2 $\bar{Q}$ 0

Estado inicial del latch: SET ($Q = 1$).

(a) Dos posibilidades para la operación SET

$\bar{S}$ 1 G_1 Q 1 0 — Se producen transiciones en la salida cuando $\bar{R}$ pasa a nivel BAJO, y permanece en el mismo estado después de que $\bar{R}$ pase a nivel ALTO.

1 0 $\bar{R}$ G_2 $\bar{Q}$ 1 0

Estado inicial del latch: SET ($Q = 1$).

$\bar{S}$ 1 G_1 Q 0

1 0 $\bar{R}$ G_2 $\bar{Q}$ 1 — No se producen transiciones porque el latch está ya en estado de RESET.

Estado inicial del latch: RESET ($Q = 0$).

(b) Dos posibilidades para la operación RESET

$\bar{S}$ 1 G_1 Q — Las salidas no cambian de estado. El latch permanece en estado SET si previamente lo estaba, o en RESET si previamente lo estaba.

$\bar{R}$ 1 G_2 $\bar{Q}$

Ambas entradas a nivel ALTO

(c) Condición de No cambio

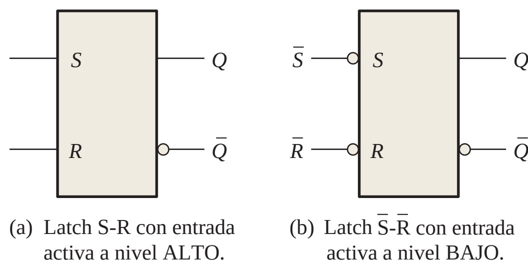
1 0 $\bar{S}$ G_1 Q 1 — Los estados de salida son indeterminados cuando se eliminan los niveles BAJOS de entrada.

1 0 $\bar{R}$ G_2 1 $\bar{Q}$

Ambas entradas a nivel BAJO

(d) Condición no válida

FIGURA 7.3 Los tres modos básicos de funcionamiento del latch $\bar{S}-\bar{R}$ (SET, RESET y No cambio) y la condición no válida.



(a) Latch S-R con entrada activa a nivel ALTO.

(b) Latch $\bar{S}$-$\bar{R}$ con entrada activa a nivel BAJO.

FIGURA 7.4 Símbolos lógicos para los latches S-R y $\bar{S}-\bar{R}$.

Entradas		Salidas		
$\overline{S}$	$\overline{R}$	Q	$\overline{Q}$	*Comentarios*
1	1	NC	NC	No cambio. El latch permanece en el estado que estaba.
0	1	1	0	Latch en estado SET.
1	0	0	1	Latch en estado RESET.
0	0	1	1	Condición no válida

TABLA 7.1 Tabla de verdad para un latch $\overline{S}-\overline{R}$ con entrada activa a nivel BAJO.

EJEMPLO 7.1

Si se aplican las formas de onda $\overline{S}$ y $\overline{R}$ de la Figura 7.5(a) a las entradas del latch de la Figura 7.4(b), determinar la forma de onda que se observará en la salida Q. Suponer que Q se encuentra inicialmente a nivel BAJO.

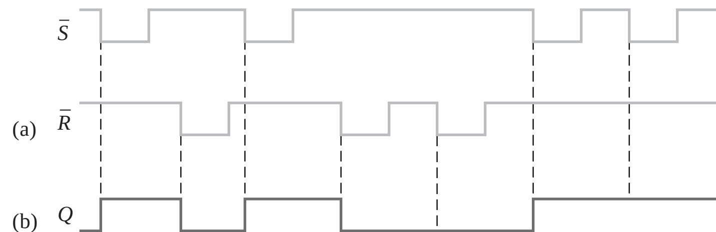


FIGURA 7.5

Solución *Véase* la Figura 7.5(b)

Problema relacionado* Determinar la salida Q de un latch S-R con entradas activas a nivel ALTO si se invierten las formas de onda de la Figura 7.5(a) y se aplican a las entradas.

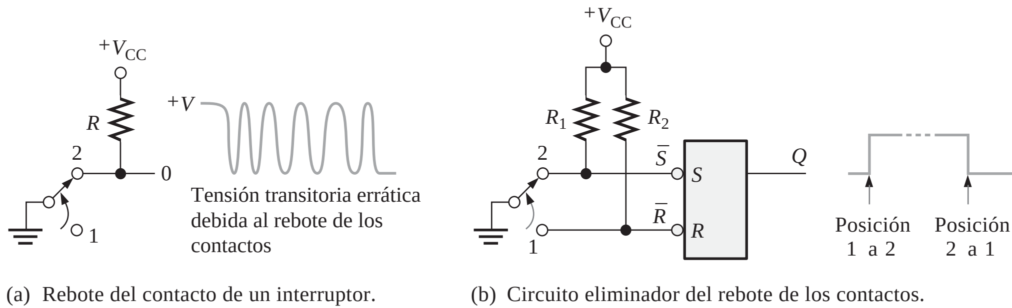
* Las respuestas se encuentran al final del capítulo.

Aplicación

El latch como eliminador del rebote de los contactos. Un buen ejemplo de aplicación de un latch $\overline{S}-\overline{R}$ consiste en la eliminación del “rebote” producido por los contactos de un interruptor mecánico. Cuando el polo de un interruptor choca con el contacto de cierre del interruptor, vibra o rebota varias veces hasta que, finalmente, se consigue un contacto firme. Aunque estos rebotes son mínimos, producen unos picos de tensión que pueden ser inadmisibles en un sistema digital. Esta situación se ilustra en la Figura 7.6(a).

Se puede utilizar un latch $\overline{S}-\overline{R}$ para eliminar los efectos de los rebotes del interruptor, como se muestra en la Figura 7.6(b). El interruptor se encuentra normalmente en la posición 1, manteniendo la entrada $\overline{R}$ a nivel BAJO y al latch en estado RESET. Cuando el interruptor pasa a la posición 2, $\overline{R}$ pasa a nivel ALTO debido a la resistencia de *pull-up* conectada a V_{CC} y $\overline{S}$ pasa a nivel BAJO cuando se produce el primer contacto. Aunque $\overline{S}$ permanece a nivel BAJO durante un breve espacio de tiempo antes de que el interruptor rebote, este tiempo es suficiente para activar (SET) el latch. Cualquier otro pico de tensión aplicado posteriormen-

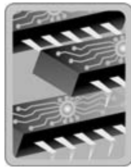
te a la entrada $\overline{S}$, debido al rebote del interruptor, no va a afectar al latch, y éste permanecerá en el estado SET. Téngase en cuenta que la salida Q del latch proporciona una transición limpia del nivel BAJO al nivel ALTO, por lo que se eliminan los picos de tensión causados por el rebote de los contactos. De forma similar, se produce una transición limpia de nivel ALTO a nivel BAJO cuando el interruptor vuelve a la posición 1.



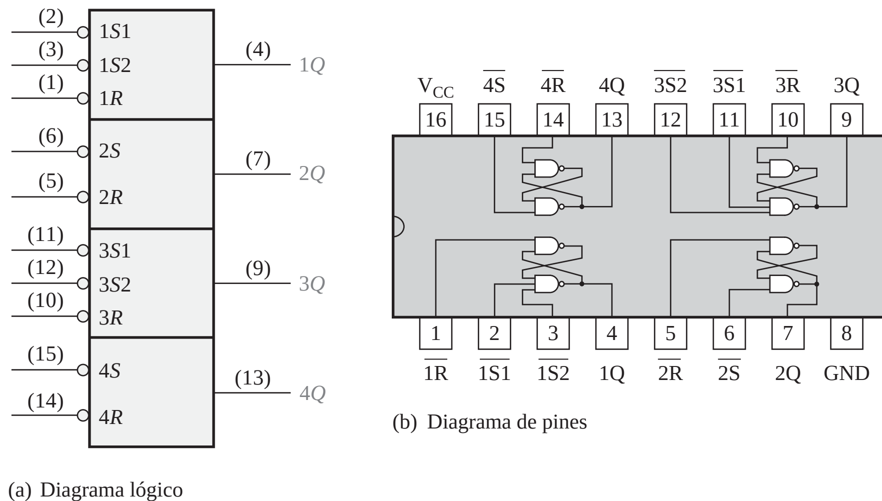
(a) Rebote del contacto de un interruptor. (b) Circuito eliminador del rebote de los contactos.

FIGURA 7.6 Utilización del latch $\overline{S}-\overline{R}$ para eliminar el rebote de los contactos de un interruptor.

LATCH SET-RESET 74LS279



El 74LS279 es un cuádruple latch $\overline{S}-\overline{R}$ representado por el diagrama lógico de la Figura 7.7(a) y cuyo diagrama de pines es el mostrado en la parte b de la misma figura. Observe que dos de los latches tienen dos entradas $\overline{S}$.



(a) Diagrama lógico

(b) Diagrama de pines

FIGURA 7.7 El cuádruple latch $\overline{S}-\overline{R}$ 74LS279.

El latch S-R con entrada de habilitación

El diagrama y el símbolo lógico de un latch con entrada de habilitación se muestran en la Figura 7.8. Las entradas S y R controlan el estado al que va a cambiar el latch cuando se aplica un nivel ALTO a la entrada

de habilitación (*EN, enable*). El latch no cambia de estado hasta que la entrada *EN* está a nivel ALTO pero, mientras que permanezca en este estado, la salida va a ser controlada por el estado de las entradas *S* y *R*. En este circuito, el estado no válido del latch se produce cuando las dos entradas *S* y *R* están simultáneamente a nivel ALTO.

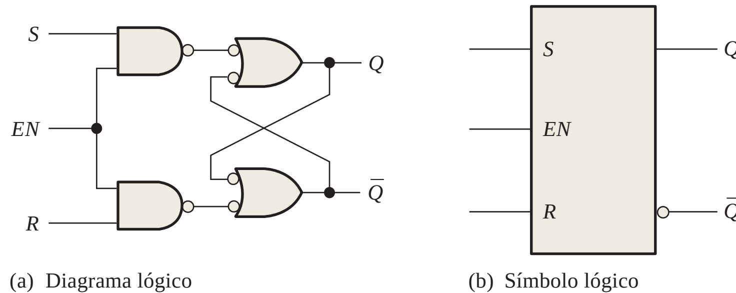


FIGURA 7.8 Latch S-R con entrada de habilitación.

EJEMPLO 7.2

Determinar la forma de onda de salida *Q,* si se aplican las señales de entrada mostradas en la Figura 7.9(a) a un latch S-R con entrada de habilitación, que se encuentra inicialmente en estado de RESET.

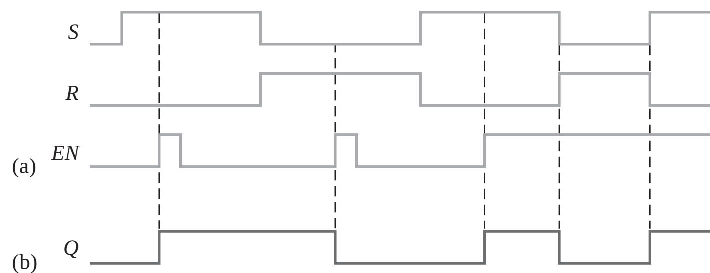


FIGURA 7.9

Solución La forma de onda *Q* se muestra en la Figura 7.9(b). Siempre que *S* está a nivel ALTO y *R* a nivel BAJO, un nivel ALTO en la entrada *EN* hace que el latch se ponga en estado SET. Siempre que *S* está a nivel BAJO y *R* a nivel ALTO, un nivel ALTO en la entrada *EN* hace que el latch se ponga en estado RESET.

Problema relacionado Determinar la salida *Q* de un latch S-R con entrada de habilitación, si se invierten las entradas *S* y *R* de la Figura 7.9(a).

El latch D con entrada de habilitación

Existe otro tipo de latch con entrada de habilitación que se denomina latch D. Se diferencia del latch S-R en que sólo tiene una entrada, además de la de habilitación, *EN*. Esta entrada recibe el nombre de entrada de datos (*D*). La Figura 7.10 muestra el diagrama y el símbolo lógico de este tipo de latch. Cuando la entrada *D* está a

nivel ALTO y la entrada *EN* también, el latch se pone en estado SET. Cuando la entrada *D* está a nivel BAJO y la entrada *EN* está a nivel ALTO, el latch se pone en estado RESET. Dicho de otra manera, la salida *Q* es igual a la entrada *D* cuando la entrada de habilitación *EN* está a nivel ALTO.

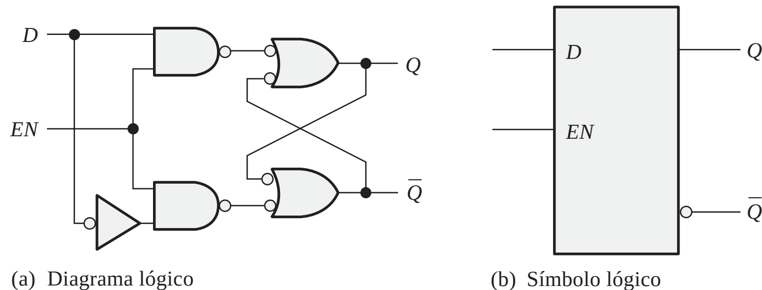


FIGURA 7.10 Latch D con entrada de habilitación.

EJEMPLO 7.3

Determinar la forma de onda de salida *Q,* si se aplican las entradas que se muestran en la Figura 7.11(a) a un latch D con entrada de habilitación que, inicialmente, está en estado RESET.

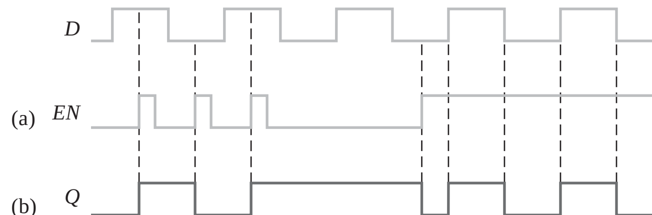
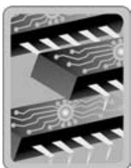


FIGURA 7.11

Solución La forma de onda *Q* se muestra en la Figura 7.11(b). Siempre que *D* y *EN* estén a nivel ALTO, la salida *Q* será un nivel ALTO. Siempre que *D* sea un nivel BAJO y *EN* esté a nivel ALTO, *Q* se pondrá a nivel BAJO. Cuando *EN* está a nivel BAJO, el estado del latch no se ve afectado por la entrada *D*.

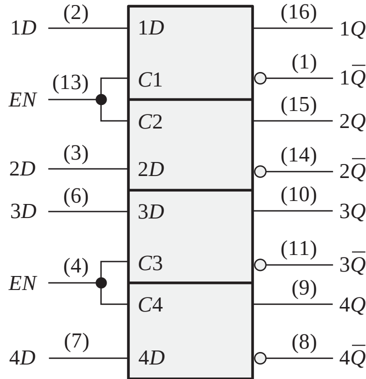
Problema relacionado Determinar la salida *Q* del latch D con entrada de habilitación, si la señal de entrada *D* de la Figura 7.11(a) se invierte.

LATCH D 74LS75



Un ejemplo de un latch D con entrada de habilitación es el 74LS75, cuyo símbolo lógico se puede ver en la Figura 7.12(a). Este dispositivo está compuesto por cuatro latches. Observe que cada entrada de habilitación *EN* activa a nivel ALTO está compartida por dos latches y se designa como entrada de control (*C*). La tabla de verdad de cada latch se muestra en la Figura 7.12(b). La X en esta tabla representa una condición “indiferente”. En este caso, cuando la entrada *EN* está a nivel BAJO, da lo mismo el valor que

tenga la entrada *D*, ya que las salidas no se ven afectadas y permanecen en los estados en que se encontraban.



(a) Símbolo lógico

Entradas		Salidas		
D	*EN*	*Q*	$\overline{Q}$	Comentarios
0	1	0	1	RESET
1	1	1	0	SET
X	0	Q_0	$\overline{Q}_0$	No cambio

Nota: Q_0 es el nivel de salida previo antes de que se establecieran las condiciones de entrada indicadas

(b) Tabla de verdad (de cada latch)

FIGURA 7.12 Cuádruple latch D con entrada de habilitación 74LS75.

REVISIÓN DE LA SECCIÓN 7.1

1. Enumerar tres tipos de latches.
2. Desarrollar la tabla de verdad del latch S-R con entradas activas a nivel ALTO de la Figura 7.1(a).
3. ¿Cuál es la salida *Q* de un latch D cuando *EN* = 1 y *D* = 1?

7.2 FLIP-FLOPS DISPARADOS POR FLANCO

Los flip-flops son dispositivos síncronos de dos estados, también conocidos como *multivibradores biestables*. En este caso, el término *síncrono* significa que la salida cambia de estado únicamente en un instante específico de una entrada de disparo denominada ***reloj*** (CLK), la cual recibe el nombre de entrada de control, *C*. Esto significa que los cambios en la salida se producen sincronizadamente con el reloj.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Definir *reloj*. ■ Definir *flip-flop disparado por flanco*. ■ Explicar la diferencia entre un flip-flop y un latch. ■ Identificar un flip-flop disparado por flanco mediante su símbolo lógico. ■ Comentar la diferencia entre los flip-flops disparados por flancos positivos y negativos. ■ Comparar el funcionamiento de los flip-flops disparados por flanco S-R, D y J-K, y explicar las diferencias entre sus tablas de verdad. ■ Explicar las entradas asíncronas de un flip-flop. ■ Describir los flip-flops 74AHC74 y 74HC112.

▲ *El indicador de entrada dinámica ▷ indica que el flip-flop cambia de estado sólo en el flanco de un pulso de reloj.*

Un **flip-flop disparado por flanco** cambia de estado con el flanco positivo (flanco de subida) o con el flanco negativo (flanco de bajada) del impulso de reloj y es sensible a sus entradas sólo en esta transición del reloj. En esta sección se cubren tres tipos de flip-flops disparados por flanco: S-R, D y J-K. Los símbolos lógicos de estos dispositivos se muestran en la Figura 7.13. Observe que pueden ser disparados por flanco positivo (no hay círculo en la

entrada C) o por flanco negativo (hay un círculo en la entrada C). La clave para identificar un flip-flop disparado por flanco mediante su símbolo lógico la da el triángulo que se encuentra dentro del bloque en la entrada del reloj (C). El triángulo se denomina *indicador de entrada dinámica*.

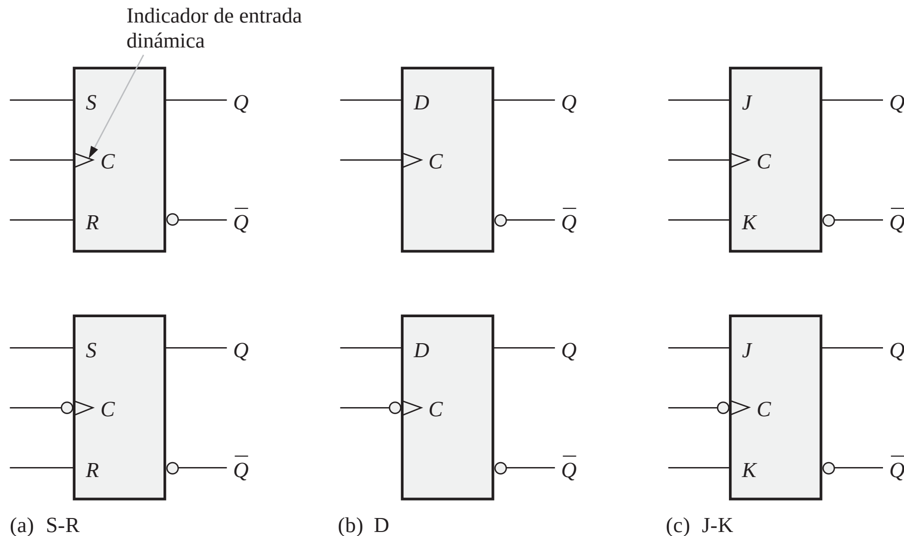


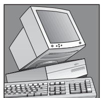
FIGURA 7.13 Símbolos lógicos de los latches disparados por flanco (parte superior: disparado por flanco positivo; parte inferior: disparado por flanco negativo).

Flip-flop S-R disparado por flanco

▲ *Un flip-flop S-R no puede tener ambas entradas R y S a nivel ALTO al mismo tiempo.*

Las entradas S y R de un **flip-flop S-R** se denominan entradas ***síncronas***, dado que los datos en estas entradas se transfieren a las salidas del flip-flop sólo con el flanco de disparo del impulso del reloj. Cuando S está a nivel ALTO y R está a nivel BAJO, la salida Q se pone a nivel ALTO con el flanco de disparo del impulso de reloj, pasando el flip-flop al estado SET. Cuando S está a nivel BAJO y R está a nivel ALTO, la salida Q se pone a nivel BAJO con el flanco de disparo del impulso de reloj, pasando el flip-flop al estado RESET. Cuando tanto S como R están a nivel BAJO, la salida no cambia de estado. Cuando S y R están a nivel ALTO, se produce una condición no válida.

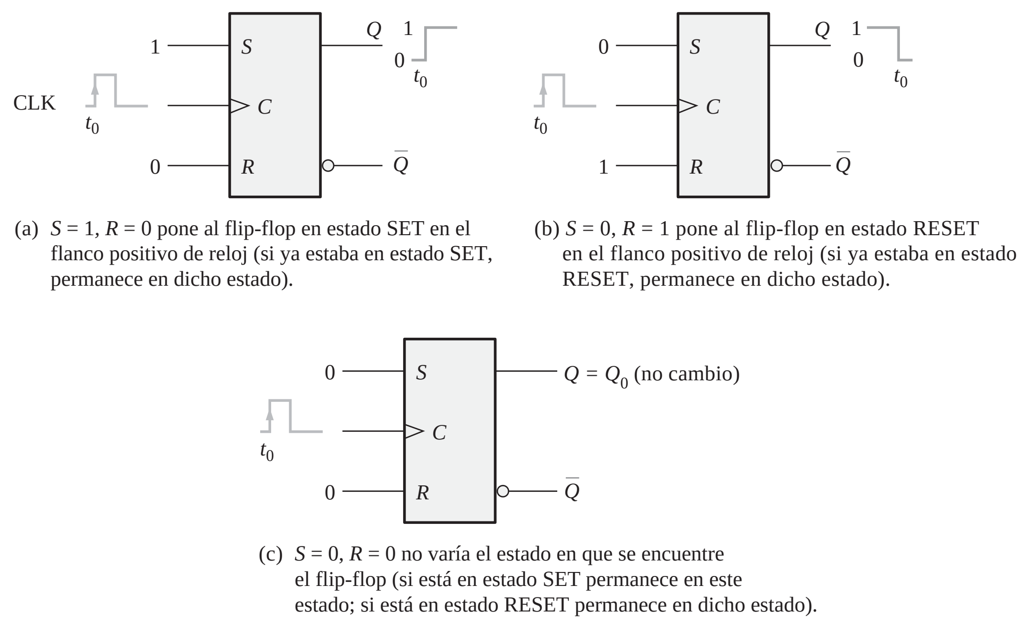
El funcionamiento básico de un flip-flop disparado por flanco positivo se muestra en la Figura 7.14, mientras que la tabla de verdad se puede ver en la Tabla 7.2. Recordemos que *un flip-flop no puede cambiar de estado excepto en el flanco de disparo de un impulso de reloj*. Las entradas S y R se pueden cambiar en cual-



NOTAS INFORMÁTICAS

Las memorias semiconductoras para computadoras constan de numerosas celdas individuales. Cada celda de almacenamiento contiene un 1 o un 0. Un tipo de memoria es la memoria de acceso aleatorio estática o SRAM, que utiliza flip-flops como celdas de almacenamiento, ya que un flip-flop mantendrá uno de dos estados de forma indefinida siempre que se aplique alimentación continua, de aquí el término *estática*. Este tipo de memoria se clasifica como una memoria *volátil*, dado que todos los datos almacenados se perderán cuando se desconecte la alimentación. Existe otro tipo de memoria, la memoria RAM dinámica o DRAM, que utiliza capacitancias en lugar de flip-flops como elemento básico de almacenamiento y debe refrescarse periódicamente para mantener los datos almacenados.

quier instante en que la entrada de reloj esté a nivel ALTO o nivel BAJO (excepto durante un breve instante de tiempo en las proximidades de las transiciones de disparo del reloj) sin que varíe la salida.



(a) $S = 1$, $R = 0$ pone al flip-flop en estado SET en el flanco positivo de reloj (si ya estaba en estado SET, permanece en dicho estado).

(b) $S = 0$, $R = 1$ pone al flip-flop en estado RESET en el flanco positivo de reloj (si ya estaba en estado RESET, permanece en dicho estado).

(c) $S = 0$, $R = 0$ no varía el estado en que se encuentre el flip-flop (si está en estado SET permanece en este estado; si está en estado RESET permanece en dicho estado).

FIGURA 7.14 Funcionamiento de un flip-flop S-R disparado por flanco positivo.

Entradas			Salidas		
S	***R***	***CLK***	***Q***	$\overline{Q}$	***Comentarios***
0	0	X	Q_0	$\overline{Q}_0$	No cambio
0	1	↑	0	1	RESET
1	0	↑	1	0	SET
1	1	↑	?	?	No válida

↑ = transición del reloj de nivel BAJO a nivel ALTO
X = irrelevante ("condición indiferente")
Q_0 = nivel de salida previo a la transición del reloj

TABLA 7.2 Tabla de verdad de un flip-flop S-R disparado por flanco positivo.

El funcionamiento y tabla de verdad de un flip-flop S-R disparado por flanco negativo son las mismas que las de un dispositivo disparado por flanco positivo, excepto en que el flanco de bajada del impulso del reloj es, en este caso, el flanco de disparo.

EJEMPLO 7.4

Determinar las formas de onda de salida Q y $\overline{Q}$ del flip-flop de la Figura 7.15, para las entradas *S, R* y *CLK* de la Figura 7.16(a). Suponer que el flip-flop disparado por flanco positivo se encuentra, inicialmente, en estado RESET.

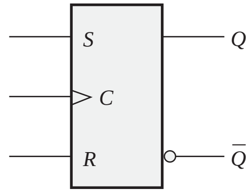


FIGURA 7.15

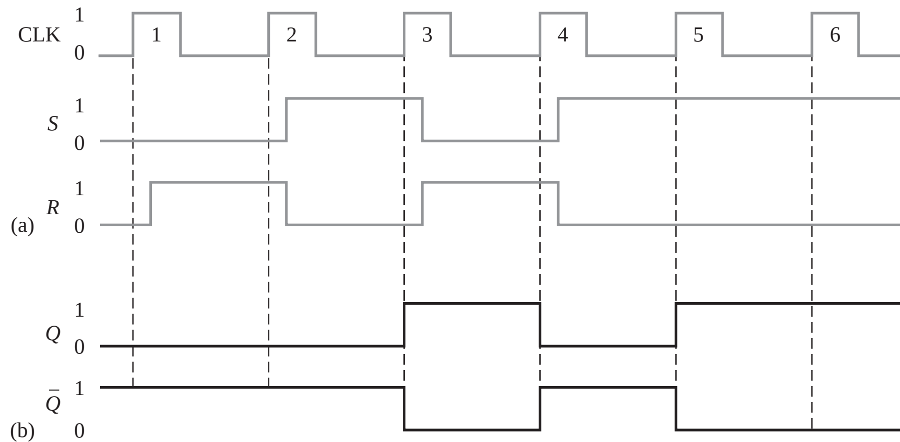


FIGURA 7.16

Solución

1. Durante el impulso 1 de reloj, *S* está a nivel BAJO y *R* está a nivel BAJO, luego *Q* no cambia.

2. Durante el impulso 2 de reloj, *S* está a nivel BAJO y *R* está a nivel ALTO, luego *Q* permanece a nivel BAJO (RESET).

3. Durante el impulso 3 de reloj, *S* está a nivel ALTO y *R* está a nivel BAJO, luego *Q* pasa a nivel ALTO (SET).

4. Durante el impulso 4 de reloj, *S* está a nivel BAJO y *R* está a nivel ALTO, luego *Q* pasa a nivel BAJO (RESET).

5. Durante el impulso 5 de reloj, *S* está a nivel ALTO y *R* está a nivel BAJO, luego *Q* pasa a nivel ALTO (SET).

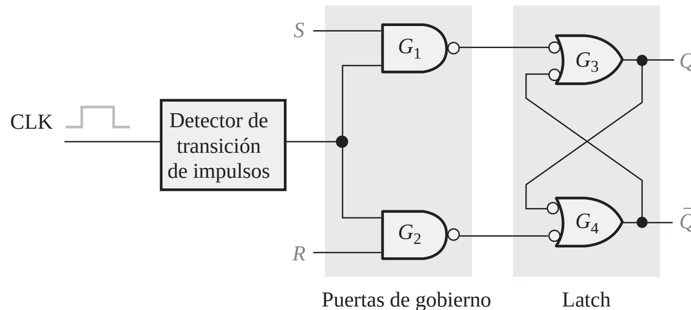
6. Durante el impulso 6 de reloj, *S* está a nivel ALTO y *R* está a nivel BAJO, luego *Q* permanece a nivel ALTO.

Una vez que se ha determinado *Q*, se puede conocer $\overline{Q}$ de forma muy sencilla, complementando la salida *Q*. Las formas de onda resultantes para *Q* y $\overline{Q}$ se muestran en la Figura 7.16(b) en función de las formas de onda de entrada de la parte (a).

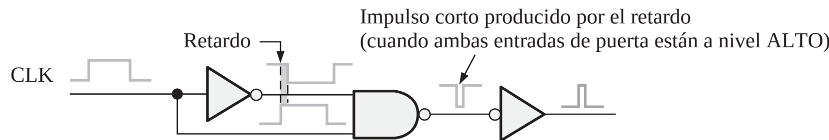
Problema relacionado Determinar *Q* y $\overline{Q}$ para las entradas *S* y *R* de la Figura 7.16(a), si el flip-flop es disparado por flanco negativo.

Un método de disparo por flanco

En la Figura 7.17(a) se presenta la implementación simplificada de un flip-flop S-R disparado por flanco, que se va a utilizar para explicar el concepto de disparo por flanco, aunque esto no significa que sea el tipo de flip-flop más importante. Realmente, los flip-flops D y J-K se utilizan con más frecuencia, y se pueden encontrar como circuitos integrados mucho más fácilmente que los S-R. Sin embargo, es mucho más interesante el estudio de los S-R, ya que tanto el flip-flop J-K como el D se pueden derivar de él. Conviene tener en cuenta que el flip-flop S-R se distingue del latch S-R con entrada de habilitación únicamente en que cuenta con un detector de transiciones de impulsos.

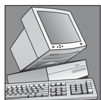


(a) Diagrama lógico simplificado de un flip-flop S-R disparado por flanco positivo



(b) Un tipo de detector de transiciones de impulsos

FIGURA 7.17 Disparo por flanco.



NOTAS INFORMÁTICAS

Todas las operaciones lógicas que se realizan mediante hardware también pueden implementarse por software. Por ejemplo, la operación de un flip-flop J-K puede realizarse mediante instrucciones específicas de computadora. Si se utilizaran dos bits para representar las entradas J y K, la computadora no haría nada para la entrada 00; para la entrada 10, un bit de datos que representara la salida Q se pondría a 1, el bit de datos Q sería 0 para la entrada 01 y el bit de datos Q se complementaría para la entrada 11. Aunque no suele ser habitual utilizar una computadora para simular un flip-flop, la cuestión es que todas las operaciones de hardware se pueden realizar mediante software.

Un detector de transiciones de impulsos típico se muestra en la Figura 7.17(b). Como se puede ver, existe un pequeño retraso en una de las entradas de la puerta NAND de manera que el impulso invertido de reloj llega a la entrada de la puerta unos cuantos nanosegundos después que el verdadero impulso de reloj. Esto origina un pico de salida que dura sólo unos nanosegundos. En los flip-flops disparados por flanco negativo, se invierte primero el impulso de reloj, de forma que se origina un pico muy estrecho en el flanco de bajada.

Observe que el circuito de la Figura 7.17 está dividido en dos secciones, una correspondiente a las denominadas puertas de gobierno, y otra al latch. Las puertas de gobierno dirigen los picos de reloj hacia la entrada de la puerta G_3 o la entrada de la G_4, dependiendo del estado de las entradas S y R. Para comprender el funcionamiento de este flip-flop, vamos a comenzar suponiendo que se encuentra en estado RESET ($Q = 0$) y que

las entradas *S, R* y *CLK* están todas a nivel BAJO. En esta situación, las salidas de las puertas G_1 y G_2 están ambas a nivel ALTO. La salida *Q* a nivel BAJO se realimenta a una de las entradas de la puerta G_4, forzando la salida $\bar{Q}$ a nivel ALTO. Puesto que $\bar{Q}$ está a nivel ALTO, las dos entradas de la puerta G_3 están a nivel ALTO (recordemos que la salida de la puerta G_1 está a nivel ALTO), manteniendo la salida *Q* a nivel BAJO. Si se aplica un impulso a la entrada de reloj CLK, las salidas de las puertas G_1 y G_2 permanecen a nivel ALTO, ya que se desactivan cuando las entradas *S* y *R* están a nivel BAJO; por tanto, no hay ningún cambio en el estado del flip-flop: permanece en RESET.

Ahora, se pone la entrada *S* a nivel ALTO, dejando *R* a nivel BAJO y se aplica un impulso de reloj. Dado que la entrada *S* de la puerta G_1 está ahora a nivel ALTO, la salida de la puerta G_1 pasa a nivel BAJO durante un breve espacio de tiempo (pico) cuando CLK pasa a nivel ALTO, haciendo que la salida *Q* se ponga a nivel ALTO. Las dos entradas de la puerta G_4 están ahora a nivel ALTO (recordemos que la salida de G_2 está a nivel ALTO ya que *R* está a nivel BAJO), forzando la salida $\bar{Q}$ a pasar a nivel BAJO. Este nivel BAJO de la salida $\bar{Q}$ se realimenta a una de las entradas de la puerta G_3, asegurando que la salida *Q* permanezca a nivel ALTO. El flip-flop se encuentra ahora en estado SET. La Figura 7.18 ilustra las transiciones de niveles lógicos que tienen lugar en el flip-flop para esta condición.

A continuación, ponemos *S* a nivel BAJO y *R* a nivel ALTO, y aplicamos un impulso de reloj. Ya que la entrada *R* está a nivel ALTO, el flanco positivo de reloj produce un pico negativo en la salida de la puerta G_2, haciendo que la salida $\bar{Q}$ pase a nivel ALTO. Debido a este nivel ALTO en $\bar{Q}$, ambas entradas de la puerta G_3 están ahora a nivel ALTO (recordemos que la salida de la puerta G_1 es un nivel ALTO debido a que la entrada *S* está a nivel BAJO), forzando a la salida *Q* a pasar a nivel BAJO. Este nivel BAJO en *Q* se realimenta a una de las entradas de la puerta G_4, asegurando así que $\bar{Q}$ permanecerá a nivel ALTO. El flip-flop se encuentra ahora en estado RESET. La Figura 7.19 ilustra las transiciones de niveles lógicos que ocurren en el flip-flop para esta condición. Al igual que en el latch con entrada de habilitación, se produce una condición no válida cuando ambas entradas *S* y *R* están, simultáneamente, a nivel ALTO. Esta es la principal desventaja de los flip-flops S-R.

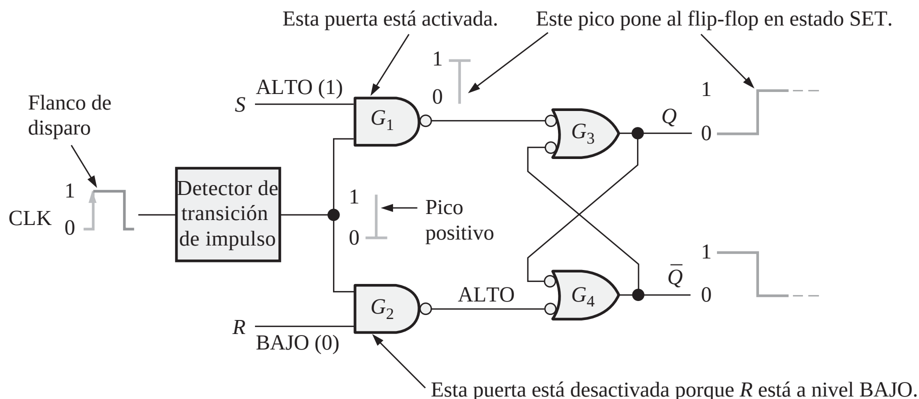


FIGURA 7.18 Flip-flop que realiza una transición del estado RESET al estado SET durante el flanco positivo del impulso del reloj.

El flip-flop D disparado por flanco

▲ *La salida Q de un flip-flop D toma el estado de la entrada D en el impulso de disparo de la señal de reloj.*

El ***flip-flop D*** resulta muy útil cuando se necesita almacenar un único bit de datos (1 o 0). Si se añade un inversor a un flip-flop S-R obtenemos un flip-flop D básico, como se muestra en la Figura 7.20, en la que se muestra uno disparado por flanco positivo.

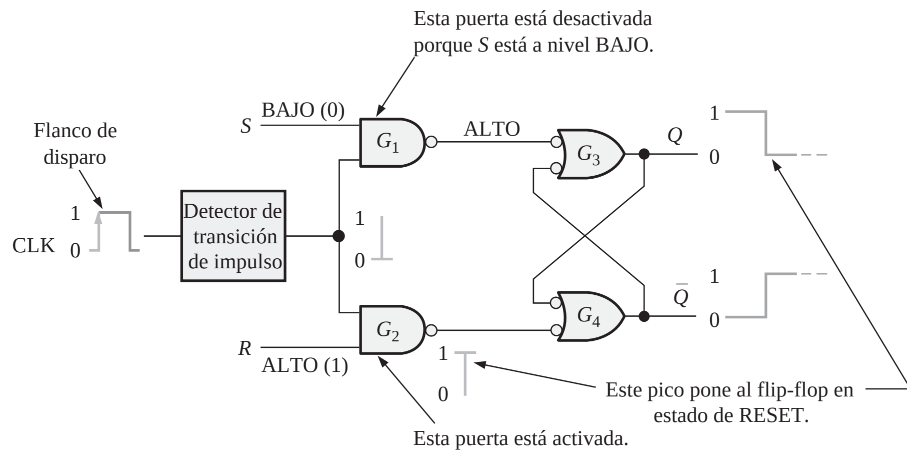


FIGURA 7.19 Flip-flop que realiza una transición del estado SET al estado RESET durante el flanco positivo del impulso del reloj.

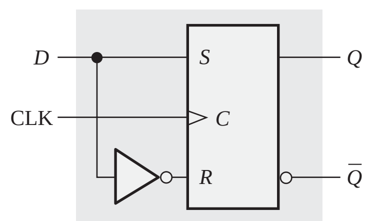


FIGURA 7.20 Flip-flop D disparado por flanco positivo, formado por un flip-flop S-R y un inversor.

Observe que el flip-flop de la Figura 7.20 tiene únicamente una entrada, la entrada *D*, además del reloj. Si cuando se aplica un impulso de reloj la entrada *D* está a nivel ALTO, el flip-flop se activa (SET) y almacena el nivel ALTO de la entrada *D* durante el flanco positivo del impulso del reloj. Si existe un nivel BAJO en la entrada *D* cuando se aplica el impulso del reloj, el flip-flop se pone a cero (RESET) y almacena el nivel BAJO de la entrada *D* durante el flanco de bajada del impulso del reloj. En el estado SET, el flip-flop almacena un 1, mientras que en el estado RESET almacena un 0.

El funcionamiento de un flip-flop D disparado por flanco positivo se resume en la Tabla 7.3. El funcionamiento de un dispositivo activado por flanco negativo es, por supuesto, idéntico, excepto que el disparo tiene lugar en el flanco de bajada del impulso del reloj. Recuerde que *Q* sigue a *D* en cada flanco del impulso de reloj.

Entradas		Salidas		
D	***CLK***	***Q***	***$\overline{Q}$***	***Comentarios***
1	↑	1	0	SET (almacena un 1)
0	↑	0	1	RESET (almacena un 0)

↑ = transición del reloj de nivel BAJO a nivel ALTO

TABLA 7.3 Tabla de verdad de un flip-flop D disparado por flanco positivo.

EJEMPLO 7.5

Dadas las formas de onda de la Figura 7.21(a) para la entrada *D* y el reloj, determinar la onda de salida *Q* si el flip-flop parte del estado RESET.

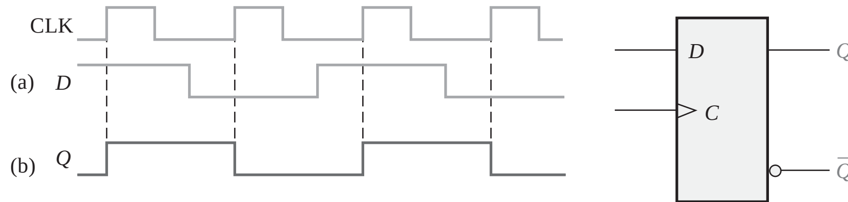


FIGURA 7.21

Solución La salida *Q* sigue al estado de la entrada *D* cada vez que se produce un flanco positivo del reloj. La salida resultante se muestra en la Figura 7.21(b).

Problema relacionado Determinar la salida *Q* para el flip-flop D, si la entrada *D* de la Figura 7.21(a) se invierte.

El flip-flop J-K disparado por flanco

El ***flip-flop J-K*** es versátil y es uno de los tipos de flip-flop más ampliamente utilizado. El funcionamiento del flip-flop J-K es idéntico al del flip-flop S-R en las condiciones de operación SET, RESET y de permanencia de estado (no cambio). La diferencia está en que el flip-flop J-K no tiene condiciones no válidas como ocurre en el S-R.

La Figura 7.22 muestra la lógica interna de un flip-flop J-K disparado por flanco positivo. Observe que se diferencia del flip-flop S-R disparado por flanco en que la salida *Q* se realimenta a la entrada de la puerta G_2, y la salida $\overline{Q}$ se realimenta a la entrada de la puerta G_1. Las dos entradas de control se denominan *J* y *K*, en honor a Jack Kilby, quien inventó el circuito integrado. Un flip-flop J-K puede ser también del tipo disparado por flanco negativo, en cuyo caso, la entrada de reloj se invierte.

Supongamos que el flip-flop de la Figura 7.23 se encuentra en estado RESET y que la entrada *J* está a nivel ALTO y la entrada *K* está a nivel BAJO. Cuando se produce un impulso de reloj, pasa un pico correspondiente al flanco anterior, indicado por ①, a través de la puerta G_1, ya que $\overline{Q}$ está a nivel ALTO y *J* también está a nivel ALTO. Esto origina que la parte latch del flip-flop cambie al estado SET. El flip-flop ahora está en estado SET.

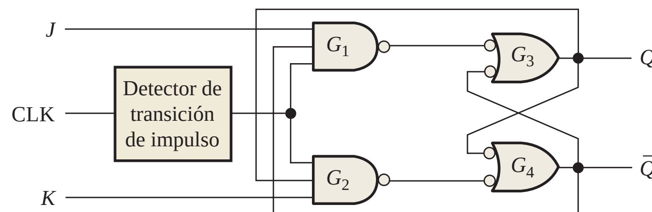


FIGURA 7.22 Diagrama lógico simplificado de un flip-flop J-K disparado por flanco positivo.

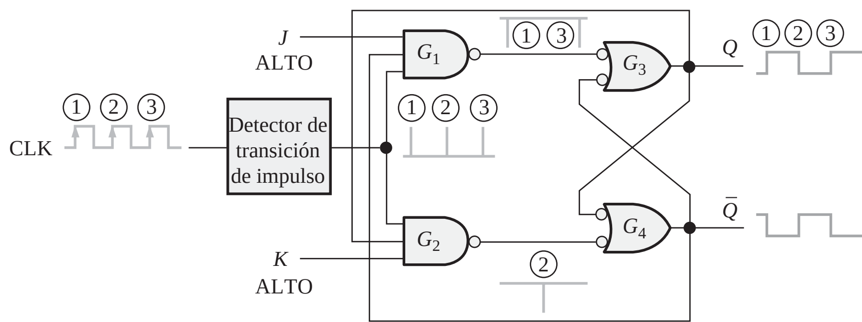


FIGURA 7.23 Transiciones que ilustran el modo de basculación cuando $J = 1$ y $K = 1$.

▲ *En el modo de basculación, un flip-flop J-K cambia de estado en cada impulso de reloj.*

Si ponemos la entrada J a nivel BAJO y la entrada K a nivel ALTO, el siguiente pico del reloj, indicado por ②, pasará a través de la puerta G_2, ya que Q está a nivel ALTO y K también. Esto origina que la parte latch del flip-flop cambie al estado RESET.

Si ahora se aplica un nivel BAJO a las dos entradas J y K, el flip-flop permanecerá en su estado actual cuando se produzca un impulso del reloj. De esta manera, un nivel BAJO en J y K origina una condición *de no cambio.*

Hasta ahora, el funcionamiento lógico del flip-flop J-K es idéntico al del S-R en sus modos SET, RESET y de no cambio. La diferencia tiene lugar cuando las dos entradas, J y K, están a nivel ALTO. Para ver esto, supongamos que el flip-flop se encuentra en estado RESET. El nivel ALTO de la salida $\overline{Q}$ activa la puerta G_1 de forma que el pico del reloj, indicado por ③, pasa y activa (SET) el flip-flop. Ahora hay un nivel ALTO en Q, el cual permite que el siguiente pico del reloj pase a través de la puerta G_2 y ponga el flip-flop en estado RESET.

Como puede ver, en cada pico sucesivo de reloj, el flip-flop cambia a su estado opuesto. A este modo de funcionamiento se le denomina **modo de *basculación* (*toggle*)**. La Figura 7.23 ilustra las transiciones cuando el flip-flop se encuentra en este modo. Un flip-flop J-K conectado en el modo de basculación en ocasiones se denomina *flip-flop T.*

En la Tabla 7.4 se muestra la tabla de verdad del flip-flop J-K disparado por flanco, la cual resume su funcionamiento. Observe que no hay ningún estado no válido, como ocurría con el flip-flop S-R. La tabla de verdad de un dispositivo disparado por flanco negativo es idéntica, excepto en que se dispara durante el flanco de bajada del impulso de reloj.

Entradas			**Salidas**		
J	***K***	***CLK***	***Q***	$\overline{Q}$	***Comentarios***
0	0	↑	Q_0	$\overline{Q}_0$	No cambio
0	1	↑	0	1	RESET
1	0	↑	1	0	SET
1	1	↑	$\overline{Q}_0$	Q_0	Basculación

↑ = transición del reloj de nivel BAJO a nivel ALTO

Q_0 = nivel de salida previo a la transición del reloj

TABLA 7.4 Tabla de verdad de un flip-flop J-K disparado por flanco positivo.

EJEMPLO 7.6

Las formas de onda de entrada de la Figura 7.24(a) se aplican a las entradas *J, K* y de reloj, tal y como se muestra. Determinar la salida *Q* suponiendo que el flip-flop se encuentra inicialmente en estado RESET.

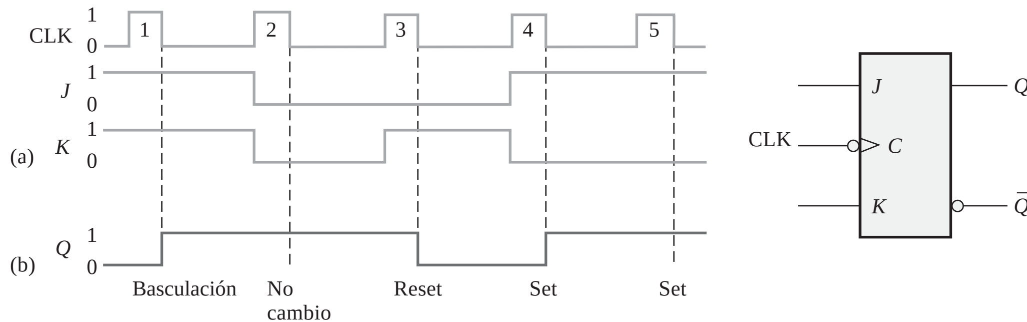


FIGURA 7.24

Solución

1. En primer lugar, dado que se trata de un flip-flop disparado por flanco negativo, como se indica mediante el círculo en la entrada de reloj, la salida *Q* cambiará sólo al ocurrir el flanco negativo del impulso de reloj.
2. En el primer impulso de reloj, *J* y *K* están a nivel ALTO y, debido a la condición de basculación, *Q* pasa a nivel ALTO.
3. En el segundo impulso de reloj, se produce la condición de no cambio en las entradas, manteniendo *Q* a nivel ALTO.
4. En el tercer impulso del reloj, *J* está a nivel BAJO y *K* a nivel ALTO, produciendo una condición de RESET, por lo que *Q* pasa a nivel BAJO.
5. En el cuarto impulso de reloj, *J* está a nivel ALTO y *K* a nivel BAJO, dando lugar a una condición de SET, luego *Q* pasa a nivel ALTO.
6. La condición SET permanece en *J* y *K* cuando ocurre el quinto impulso del reloj, de forma que *Q* sigue a nivel ALTO.

La forma de onda *Q* resultante se indica en la Figura 7.24(b).

Problema relacionado Determinar la salida *Q* del flip-flop J-K si las entradas *J* y *K* de la Figura 7.24(a) se invierten.

EJEMPLO 7.7

Las formas de onda de la Figura 7.25(a) se aplican al flip-flop que se muestra. Determinar la salida *Q*, comenzado en el estado RESET.

Solución La salida *Q* toma el estado determinado por los estados de las entradas *J* y *K* en el flanco positivo (flanco de disparo) del impulso de reloj. Si se produce un cambio en *J* o en *K* después del disparo del reloj éste no tiene efecto en la salida, como se muestra en la Figura 7.25(b).

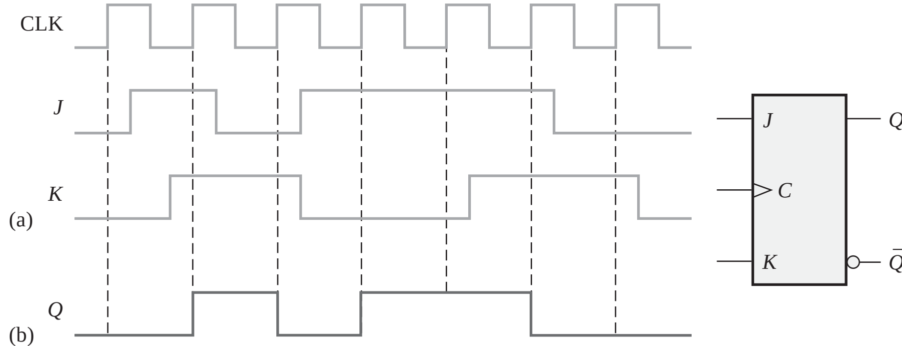


FIGURA 7.25

Problema relacionado Intercambiar las entradas *J* y *K* y determinar la salida *Q* resultante.

Entradas asíncronas de inicialización y borrado

▲ *Una entrada de inicialización activa pone la salida Q a nivel ALTO (SET).*

▲ *Una entrada de borrado activa pone la salida Q a nivel BAJO (RESET).*

En los flip-flops que acabamos de estudiar, el *S-R*, el *D* y el *J-K*, se dice que sus entradas son *entradas síncronas,* ya que los datos de estas entradas condicionan la salida de los flip-flops sólo durante el flanco de disparo del impulso de reloj; esto significa que los datos se transfieren sincronizados con la señal de reloj.

La mayoría de los circuitos integrados flip-flops tienen también entradas **asíncronas**. Estas son entradas que pueden variar el estado del flip-flop *independientemente del reloj*. Generalmente, los fabricantes las denominan de ***inicialización*, *preset*,** (*PRE)* y ***borrado, clear*,** (*CLR*), o de *activación directa* ($S_{D,}$ direct SET) y *desactivación directa* (R_{D}, direct RESET). Un nivel activo en la entrada de inicialización del flip-flop (preset) pone a SET el dispositivo, y un nivel activo en la entrada de borrado (*clear*) lo pone en estado RESET. En la Figura 7.26 se muestra el símbolo lógico de un flip-flop J-K con entradas *preset* y *clear*. Estas entradas son activas a nivel BAJO, como indican los círculos. Estas entradas de inicialización y borrado deben mantenerse a nivel ALTO para el funcionamiento síncrono.

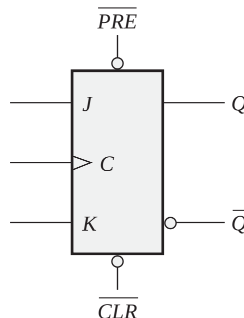


FIGURA 7.26 Símbolo lógico de un flip-flop J-K con entrada de inicialización (*preset*) y de borrado (*clear*) activas a nivel BAJO.

La Figura 7.27 muestra el diagrama lógico de un flip-flop J-K disparado por flanco con entradas de inicialización y borrado activas a nivel BAJO ($\overline{PRE}$) y ($\overline{CLR}$). Esta figura ilustra, básicamente, cómo funcionan estas entradas. Como puede ver, están conectadas de forma que anulan el efecto de las entradas síncronas *J*, *K* y el reloj.

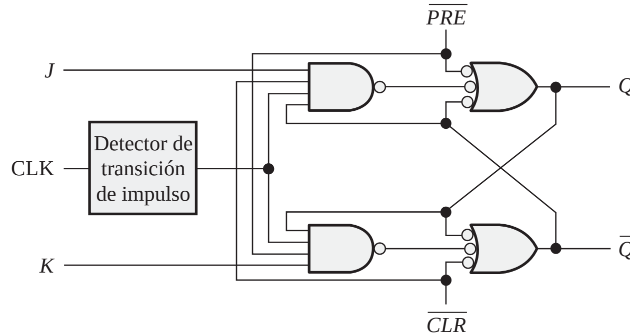


FIGURA 7.27 Diagrama lógico de un flip-flop J-K con entradas de inicialización (*preset*) y de borrado (*clear*) activas a nivel BAJO.

EJEMPLO 7.8

En el flip-flop J-K activado por flanco positivo de la Figura 7.28, con entradas *preset* y *clear*, determinar la salida *Q* para las entradas mostradas en el diagrama de tiempos de la parte (a), si *Q* está inicialmente a nivel BAJO.

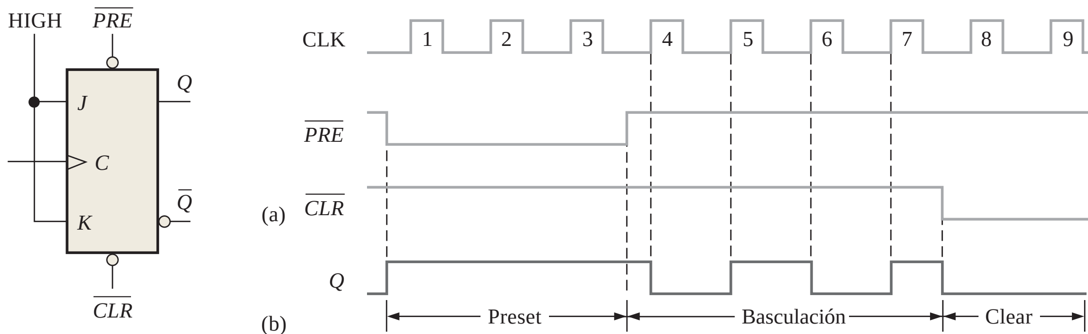


FIGURA 7.28

Solución

1. Durante los impulsos de reloj 1, 2 y 3, la entrada de inicialización ($\overline{PRE}$) está a nivel BAJO, manteniendo el flip-flop en estado SET, independientemente de las entradas síncronas *J* y *K*.
2. Durante los impulsos 4, 5, 6 y 7, funciona en modo de basculación, dado que *J* está a nivel ALTO, *K* está a nivel ALTO y tanto $\overline{PRE}$ como $\overline{CLR}$ están a nivel ALTO.

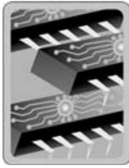
3. Para los impulsos de reloj 8 y 9, la entrada de borrado ($\overline{CLR}$) está a nivel BAJO, por lo que el flip-flop se mantiene en estado RESET, independientemente de las entradas síncronas.

La salida Q resultante se muestra en la Figura 7.28(b).

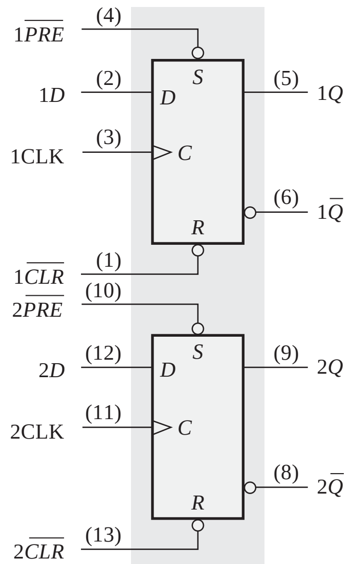
Problema relacionado Si intercambiamos las formas de onda de entrada $\overline{PRE}$ y $\overline{CLR}$ de la Figura 7.28(a), ¿qué forma tendrá la onda de salida Q?

Ahora, se van a tratar dos circuitos integrados flip-flops disparados por flanco, que son representativos de varios tipos de flip-flops disponibles en forma de CI y que, al igual que la mayoría de otros dispositivos, se encuentran disponibles en las familias lógicas TTL y CMOS.

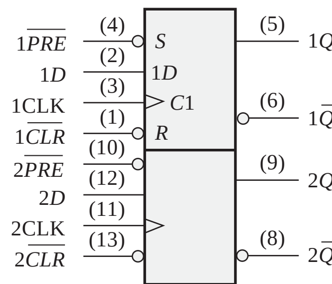
DOBLE FLIP-FLOP D 74AHC74



Este dispositivo CMOS contiene dos flip-flops D idénticos que son independientes entre sí, excepto en que comparten V_{CC} y tierra. Son flip-flops disparados por flanco positivo y disponen de las entradas asíncronas de inicialización y borrado activas a nivel BAJO. En la Figura 7.29(a) se muestran los símbolos lógicos de cada flip-flop individual dentro del encapsulado, mientras que en la parte (b) de la figura podemos ver el símbolo estándar ANSI/IEEE, que representa el dispositivo completo. La numeración de los pines se indica entre paréntesis.



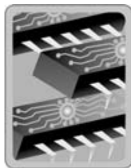
(a) Símbolos lógicos individuales



(b) Bloque lógico
Nota: La S y la R de dentro del bloque indican que $\overline{PRE}$ pone al dispositivo en estado SET y $\overline{CLR}$ le pone en estado RESET.

FIGURA 7.29 Símbolos lógicos del doble flip-flop D disparado por flanco positivo 74AHC74.

DOBLE FLIP-FLOP J-K 74HC112



Este dispositivo CMOS contiene también dos flip-flops idénticos que son disparados por flanco negativo, y tienen entradas asíncronas de inicialización y de borrado activas a nivel BAJO. Los símbolos lógicos correspondientes se muestran en la Figura 7.30.

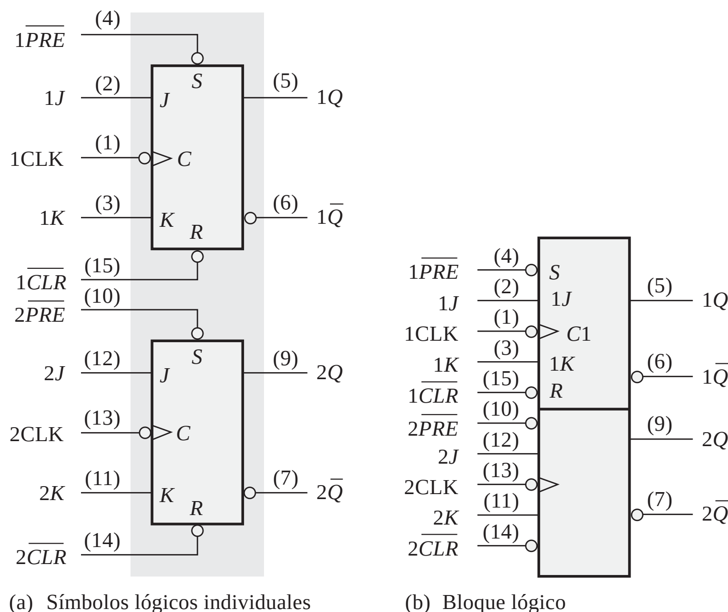


FIGURA 7.30 Símbolos lógicos del doble flip-flop J-K disparado por flanco negativo 74HC112.

EJEMPLO 7.9

Las formas de onda para 1*J*, 1*K*, 1CLK, $1\overline{PRE}$ y $1\overline{CLR}$ de la Figura 7.31(a) se aplican a uno de los flip-flops disparados por flanco negativo del circuito 74HC112. Determinar la onda de salida 1*Q*.

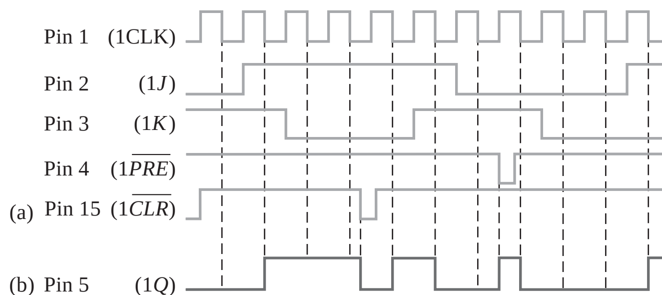


FIGURA 7.31

Solución	La forma de onda resultante $1Q$ se muestra en la Figura 7.31(b). Observe que cada vez que se aplica un nivel BAJO a la entrada $1\overline{PRE}$ o a la $1\overline{CLR}$, el flip-flop pasa a estado SET o RESET independientemente del estado del resto de las entradas.
Problema relacionado	Determinar la forma de onda de salida $1Q$ si se intercambian las señales $1\overline{PRE}$ y $1\overline{CLR}$.

REVISIÓN DE LA SECCIÓN 7.2

1. Describir la principal diferencia entre un latch S-R con entrada de habilitación y un flip-flop S-R disparado por flanco.
2. ¿Cuál es la diferencia en el funcionamiento básico entre un flip-flop J-K y un flip-flop S-R?
3. Suponer que el flip-flop de la Figura 7.21 es del tipo disparado por flanco negativo. Describir la forma de onda de salida, para las mismas señales de reloj y datos (CLK y D).

7.3 CARACTERÍSTICAS DE OPERACIÓN DE LOS FLIP-FLOPS

El funcionamiento, requisitos de operación y limitaciones de los flip-flops se especifican mediante varias características de funcionamiento o parámetros que se encuentran en las hojas de características del dispositivo. Generalmente, las especificaciones son aplicables a todos los flip-flops CMOS y TTL.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Definir *retardo de propagación*. ■ Explicar las distintas especificaciones de retardos de propagación. ■ Definir *tiempo de establecimiento* y explicar en qué limita el funcionamiento de los flip-flops. ■ Definir *tiempo de mantenimiento* y explicar en qué limita el funcionamiento de los flip-flops. ■ Explicar el significado de la frecuencia máxima de reloj. ■ Explicar las distintas especificaciones de los anchos de los impulsos. ■ Definir *disipación de potencia* y calcular su valor en un dispositivo determinado. ■ Comparar varias series de flip-flops en función de sus parámetros de funcionamiento.

Retardos de propagación

Se define ***retardo de propagación*** como el intervalo de tiempo requerido para que se produzca un cambio en la salida una vez que se ha aplicado una señal en la entrada. Existen distintas categorías de retardos de propagación que son importantes en el funcionamiento de los flip-flops:

1. El retardo de propagación t_{PLH} se mide desde el flanco de disparo del impulso de reloj hasta la transición de nivel BAJO a nivel ALTO de la salida. Este retardo se ilustra en la Figura 7.32(a).
2. El retardo de propagación t_{PHL} se mide desde el flanco de disparo de impulso del reloj hasta la transición de nivel ALTO a nivel BAJO de la salida. Este retardo se ilustra en la Figura 7.32(b).
3. El retardo de propagación t_{PLH} medido desde la entrada de inicialización (*preset*) hasta la transición de nivel BAJO a nivel ALTO de la salida. Este retardo se ilustra en la Figura 7.33(a), para una entrada de inicialización activa a nivel BAJO.
4. El retardo de propagación t_{PHL} medido desde la entrada de borrado (*clear*) hasta la transición de nivel ALTO a nivel BAJO de la salida. Este retardo se ilustra en la Figura 7.33(b), para una entrada de borrado activa a nivel BAJO.

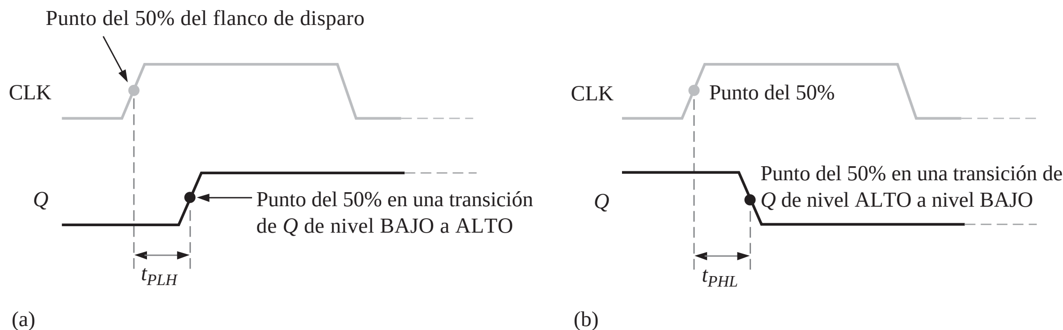


FIGURA 7.32 Retardos de propagación entre el reloj y la salida.

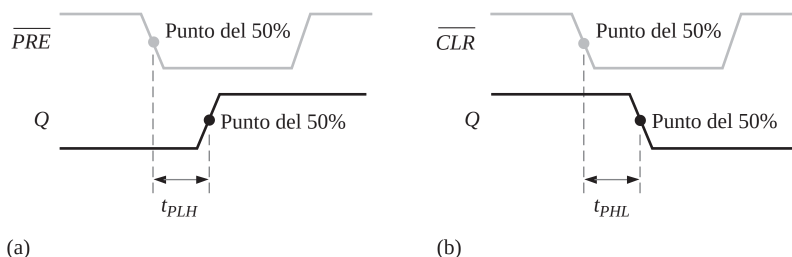


FIGURA 7.33 Retardos de propagación entre la entrada de inicialización y la salida y entre la entrada de borrado y la salida.

Tiempo de establecimiento

El ***tiempo de establecimiento,*** *setup time* (t_S) es el intervalo mínimo que los niveles lógicos deben mantener constantes en las entradas (*J* y *K*, *S* y *R* o *D*) antes de que llegue el flanco de disparo del impulso de reloj, de modo que dichos niveles sincronicen correctamente en el flip-flop. Este intervalo, para el caso de un flip-flop D, se muestra en la Figura 7.34.

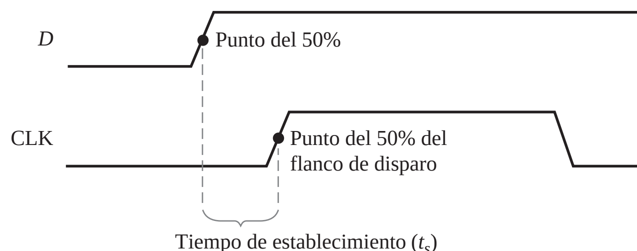


FIGURA 7.34 Tiempo de establecimiento (t_s). El nivel lógico debe estar presente en la entrada *D* durante un período de tiempo igual o mayor que t_s antes de que el flanco de disparo del impulso de reloj para tener una entrada de datos correcta.

Tiempo de mantenimiento

El ***tiempo de mantenimiento***, *hold time* (t_h) es el intervalo mínimo que los niveles lógicos deben mantenerse constantes en las entradas después de que haya pasado el flanco de disparo del impulso de reloj, de modo que

dichos niveles se sincronicen correctamente en el flip-flop. Esto se ilustra, para el caso de un flip-flop D, en la Figura 7.35.

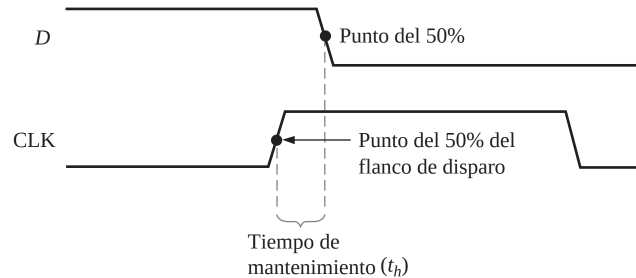


FIGURA 7.35 Tiempo de mantenimiento (t_h). El nivel lógico debe permanecer en la entrada *D* durante un período de tiempo igual o mayor que t_h después de que el flanco de disparo del impulso de reloj para tener una entrada de datos correcta.

Frecuencia máxima de reloj

La frecuencia máxima de reloj ($f_{máx}$) es la mayor velocidad a la que se puede disparar el flip-flop de manera fiable. Para frecuencias del reloj por encima de la máxima, el flip-flop puede ser incapaz de responder lo suficientemente rápido y su funcionamiento se vería deteriorado.

Anchura de los impulsos

Usualmente, los fabricantes especifican la anchura mínima de los impulsos (t_W) para un funcionamiento adecuado de las entradas de reloj, inicialización y borrado. Típicamente, el reloj se especifica mediante sus intervalos de tiempo mínimo para los niveles ALTO y BAJO.

Disipación de potencia

La ***disipación de potencia*** de cualquier circuito digital se define como la potencia total consumida por el dispositivo. Por ejemplo, si el flip-flop funciona con una fuente de continua de +5 V y circula por él una corriente de 5 mA, la disipación de potencia es:

$$P = V_{CC} \times I_{CC} = 5\text{ V} \times 5\text{ mA} = 25\text{ mW}$$

Esta disipación de potencia es muy importante en la mayoría de las aplicaciones en las que la capacidad de la fuente de continua (dc) juegue un papel importante. Como ejemplo, vamos a suponer que tenemos un sistema digital que requiere un total de diez flip-flops, y que cada uno de ellos disipa una potencia de 25 mW. El requisito de disipación de potencia total es:

$$P_T = 10 \times 25\text{ mW} = 250\text{ mW} = 0{,}25\text{ W}$$

Esto nos dice cuál es la potencia de salida necesaria de nuestra fuente de alimentación. Si los flip-flops funcionan con +5 V de continua, entonces la corriente total que tiene que suministrar la fuente es la siguiente:

$$I = \frac{250\text{ mW}}{5\text{ V}} = 50\text{ mA}$$

Tenemos que utilizar una fuente de +5 V que sea capaz de proporcionar al menos 50 mA de corriente.

Comparación de flip-flops específicos

La Tabla 7.5 proporciona una comparación para cuatro flip-flops TTL y CMOS del mismo tipo, en función de los distintos parámetros de funcionamiento discutidos en esta sección.

Parámetro	**CMOS**		**TTL**	
	74HC74A	**74AHC74**	**74LS74A**	**74F74**
t_{PHL} (CLK a Q)	17 ns	4,6 ns	40 ns	6,8 ns
t_{PLH} (CLK a Q)	17 ns	4,6 ns	25 ns	8,0 ns
t_{PHL} ($\overline{CLR}$ a Q)	18 ns	4,8 ns	40 ns	9,0 ns
t_{PLH} ($\overline{PRE}$ a Q)	18 ns	4,8 ns	25 ns	6,1 ns
t_S (tiempo de setup)	14 ns	5,0 ns	20 ns	2,0 ns
t_h (tiempo de hold)	3,0 ns	0,5 ns	5 ns	1,0 ns
t_W (CLK HIGH)	10 ns	5,0 ns	25 ns	4,0 ns
t_W (CLK LOW)	10 ns	5,0 ns	25 ns	5,0 ns
t_W ($\overline{CLR}/\overline{PRE}$)	10 ns	5,0 ns	25 ns	4,0 ns
$f_{máx}$ (MHz)	35 MHz	170 MHz	25 MHz	100 MHz
Potencia (mW)	0,012 mW	1,1 mW		
Potencia (mW), ciclo de trabajo 50%			44 mW	88 mW

TABLA 7.5 Comparación de los parámetros de funcionamiento para cuatro familias de CI flip-flop del mismo tipo a 25 ºC.

CONSEJOS PRÁCTICOS

Una ventaja de los dispositivos CMOS es que pueden operar en un más amplio rango de tensiones continuas de alimentación (normalmente de 2 V a 6 V) que los dispositivos TTL, y, por tanto, se pueden emplear fuentes de alimentación más baratas que no necesitan una regulación precisa. Para los circuitos CMOS también se puede utilizar baterías como fuentes de alimentación primarias o secundarias. Además, tensiones más bajas significan que el CI disipa menos potencia. El inconveniente es que el rendimiento de un circuito CMOS se degrada con tensiones de alimentación bajas. Por ejemplo, la frecuencia máxima de reloj garantizada de un flip-flop CMOS es mucho menor para V_{CC} = 2 V que para V_{CC} = 6 V.

REVISIÓN DE LA SECCIÓN 7.3

1. Definir los siguientes parámetros:
(a) tiempo de establecimiento **(b)** tiempo de mantenimiento

2. ¿Cuál de todos los flip-flops de la Tabla 7.5 puede funcionar a mayor frecuencia?

7.4 APLICACIONES DE LOS FLIP-FLOPS

En esta sección, se describen tres aplicaciones de carácter general de los flip-flops que nos van a proporcionar una idea básica de cómo pueden utilizarse. En los Capítulos 8 y 9 se tratarán en más detalle las aplicaciones de los flip-flops en contadores y registros.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Explicar la aplicación de los flip-flops en el almacenamiento de datos. ■ Describir cómo se emplean los flip-flops para la división de frecuencia. ■ Explicar cómo se usan los flip-flops en aplicaciones básicas de contadores.

Almacenamiento de datos paralelo

Uno de los requisitos más comunes de los sistemas digitales consiste en almacenar de forma simultánea una serie de bits de datos, procedentes de varias líneas paralelas, en un grupo de flip-flops. Este proceso se ilustra en la Figura 7.36(a), utilizando cuatro flip-flops. Cada una de las cuatro líneas paralelas de datos se conecta a la entrada *D* de un flip-flop. Las entradas de reloj de los flip-flops se conectan juntas, de forma que los flip-flops son disparados mediante el mismo impulso del reloj. En este ejemplo, se utilizan flip-flops disparados por flanco positivo, por lo que los datos de las entradas *D* se almacenan simultáneamente en los flip-flops con el flanco positivo de reloj, como se indica en el diagrama de tiempos de la Figura 7.36(b). Además, las entradas de puesta a cero asíncronas (*R*) se conectan a una línea $\overline{CLR}$ común, que inicialmente pone a cero a todos los flip-flops.

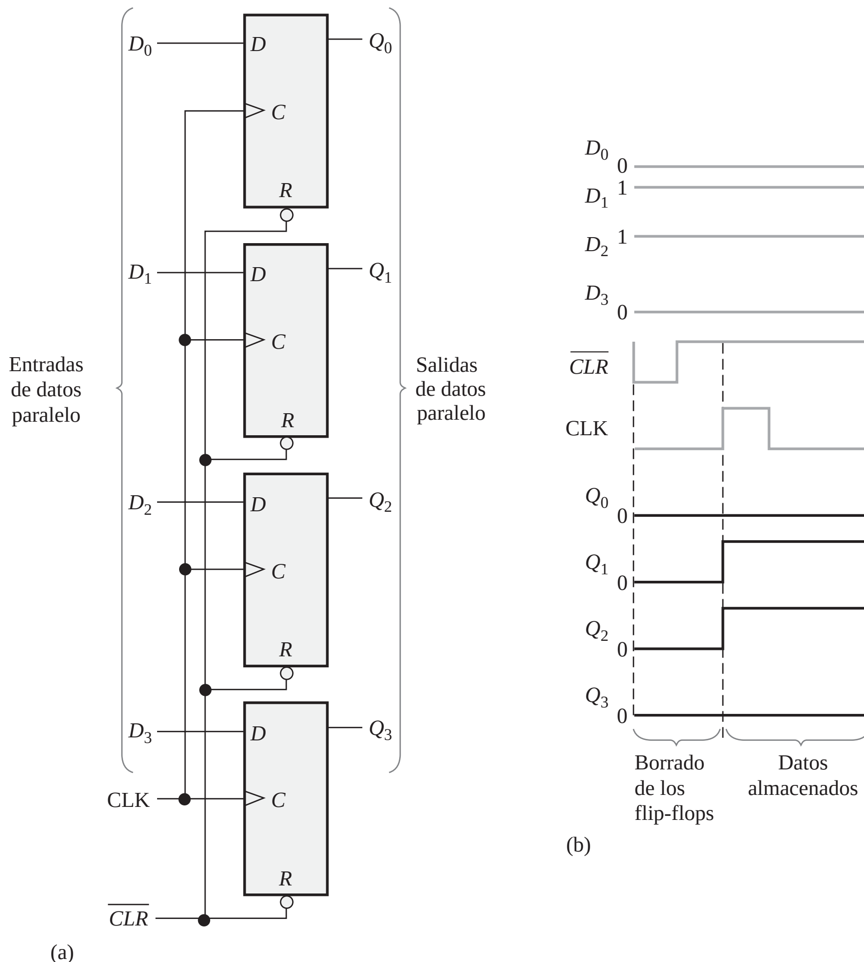


FIGURA 7.36 Ejemplo de utilización de flip-flops en un registro básico para almacenamiento paralelo de datos.

Este grupo de cuatro flip-flops es un ejemplo de un registro básico utilizado para almacenamiento de datos. En los sistemas digitales, los datos se almacenan normalmente en grupos de bits (usualmente ocho o múltiplos de ocho), que representa números, códigos u otras informaciones. Los registros se tratarán en detalle en el Capítulo 9.

División de frecuencia

Otra de las aplicaciones de un flip-flop es la división (reducción) de frecuencia de una señal periódica. Cuando se aplica un tren de impulsos a la entrada de reloj de un flip-flop J-K conectado en modo de basculación ($J = K = 1$), la salida Q es una señal cuadrada que tiene una frecuencia igual a la mitad de la que tiene la señal de reloj. Por tanto, se puede utilizar un único flip-flop como un divisor por 2, como muestra la Figura 7.37. Como puede verse, el flip-flop cambia de estado en cada flanco de disparo del impulso de reloj (flancos positivos en este caso). Esto da lugar a una salida que varía a la frecuencia mitad de la señal de reloj.

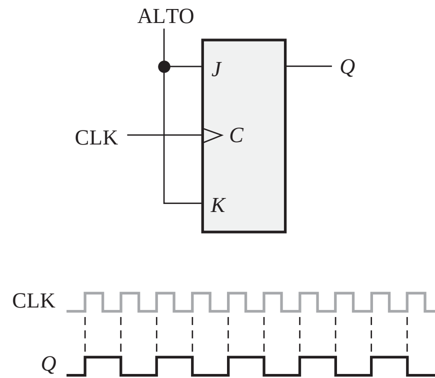


FIGURA 7.37 El flip-flop J-K como dispositivo divisor por 2. La frecuencia de Q es la frecuencia mitad de la señal *CLK*.

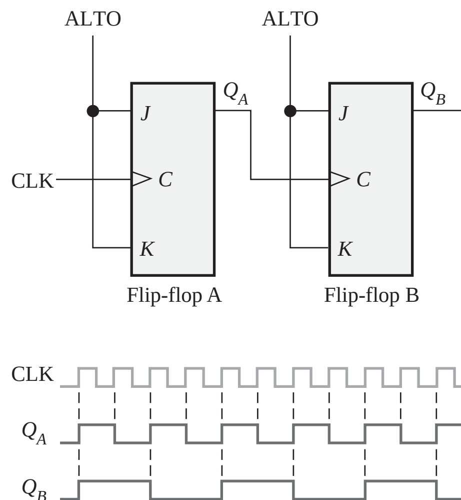


FIGURA 7.38 Ejemplo de utilización de dos flip-flops J-K para dividir la frecuencia de reloj por 4. La frecuencia de Q_A es la frecuencia mitad de *CLK* y la frecuencia de Q_B es un cuarto de la frecuencia de *CLK*.

Se pueden conseguir divisiones sucesivas de la frecuencia del reloj conectando la salida de un flip-flop a la entrada de reloj de un segundo flip-flop, como se muestra en la Figura 7.38. El flip-flop B divide la frecuencia de la salida Q_A por 2. La salida Q_B es, por tanto, un cuarto de la frecuencia de la señal de reloj original. En estos diagramas de tiempo no se muestran los retardos de propagación.

Si se conectan varios flip-flops de esta manera, se puede conseguir una división de frecuencias de 2^n, donde n es el número de flip-flops. Por ejemplo, tres flip-flops dividen la frecuencia de reloj por $2^3 = 8$; cuatro flip-flops dividen la frecuencia de reloj por $2^4 = 16$, y así sucesivamente.

EJEMPLO 7.10

Desarrollar la forma de onda f_{out} para el circuito de la Figura 7.39, cuando se aplica una señal cuadrada de 8 kHz en la entrada de reloj del flip-flop A.

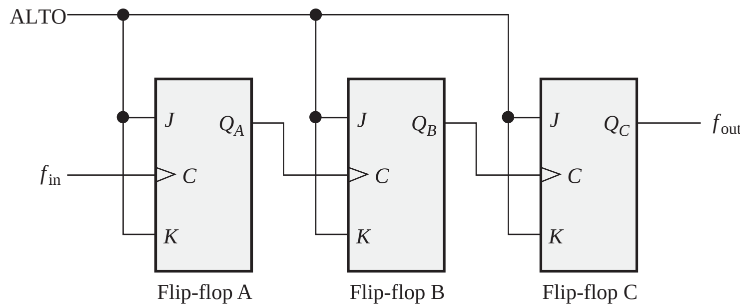


FIGURA 7.39

Solución Los tres flip-flops están conectados para dividir la frecuencia de entrada por ocho ($2^3 = 8$) y la señal f_{out} se muestra en la Figura 7.40. Dado que se trata de flip-flops disparados por flanco positivo, las salidas cambian durante el flanco positivo del reloj. Hay un impulso de salida por cada ocho impulsos de entrada, de forma que la frecuencia de salida es 1 kHz. Las señales Q_A y Q_B también se muestran.

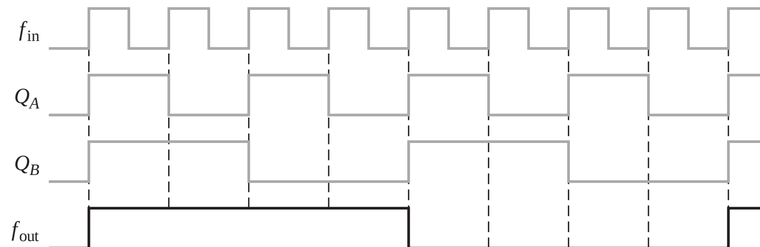


FIGURA 7.40

Problema relacionado ¿Cuántos flip-flops se requieren para dividir una frecuencia entre treinta y dos?

Contadores

Otra de las aplicaciones importantes de los flip-flops son los contadores digitales, que serán tratados en detalle en el Capítulo 8. El concepto se ilustra en la Figura 7.41. Los flip-flops son de tipo J-K disparados por flanco negativo. Ambos flip-flops se encuentran inicialmente en estado RESET. El flip-flop A bascula en las transiciones negativas de cada impulso de reloj. La salida Q del flip-flop A dispara el flip-flop B, de manera que siempre que Q_A realiza una transición de nivel ALTO a nivel BAJO, el flip-flop B bascula. Las señales resultantes Q_A y Q_B se muestran en la figura.

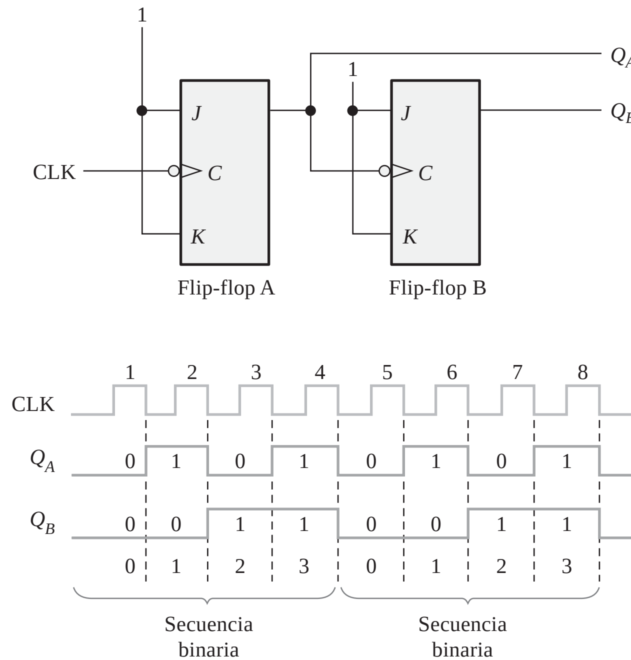


FIGURA 7.41 Flip-flops utilizados para generar una secuencia de cuenta binaria. Se muestran dos repeticiones (00, 01, 10, 11).

Observe la secuencia de Q_A y Q_B en la Figura 7.41. Previamente al impulso de reloj 1, $Q_A = 0$ y $Q_B = 0$; tras el impulso de reloj 1, $Q_A = 1$ y $Q_B = 0$; después del impulso de reloj 2, $Q_A = 0$ y $Q_B = 1$, y tras el impulso de reloj 3, $Q_A = 1$ y $Q_B = 1$. Si se toma Q_A como el bit menos significativo, se produce una secuencia binaria de dos bits a medida que se disparan los flip-flops. Esta secuencia binaria se repite cada cuatro impulsos de reloj, como se muestra en el diagrama de tiempos de la Figura 7.41. Por tanto, los flip-flops siguen una secuencia de 0 a 3 (00, 01, 10, 11) y luego vuelven a 0 para comenzar la misma secuencia de nuevo.

EJEMPLO 7.11

Determinar las formas de onda de salida en función del reloj para Q_A, Q_B y Q_C en el circuito de la Figura 7.42 y mostrar la secuencia binaria representada por estas señales.

Solución El diagrama de tiempos de salida se muestra en la Figura 7.43. Observe que las salidas cambian en los flancos negativos de los impulsos de reloj. Las salidas siguen la secuencia binaria 000, 001, 010, 011, 100,101, 110 y 111, tal y como se indica.

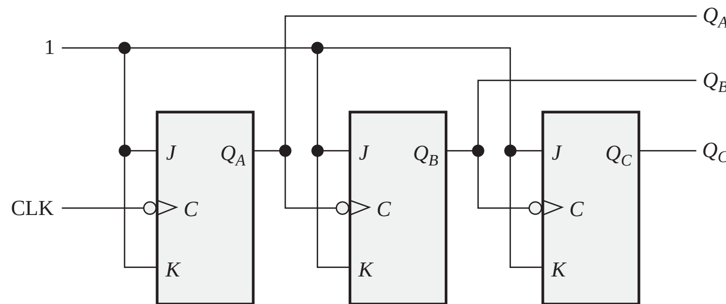


FIGURA 7.42

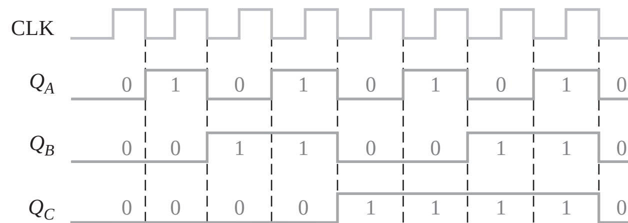


FIGURA 7.43

Problema relacionado ¿Cuántos flip-flops son necesarios para generar una secuencia binaria que represente los números decimales de 0 a 15?

REVISIÓN DE LA SECCIÓN 7.4

1. ¿Cómo se denomina un conjunto de flip-flops utilizado para almacenamiento de datos?
2. ¿Cómo se tiene que conectar un flip-flop J-K para funcionar como un dispositivo divisor por 2?
3. ¿Cuántos flip-flops son necesarios para obtener un dispositivo divisor por 64?

7.5 MONOESTABLES

Los ***monoestables*** son dispositivos multivibradores que sólo tienen un único estado estable. Normalmente, un monoestable se encuentra en su estado estable, cambiando a su estado inestable sólo cuando se dispara. Una vez que se ha disparado, el monoestable permanece en su estado inestable durante un determinado intervalo de tiempo, volviendo a continuación a su estado estable. El tiempo que este dispositivo permanece en el estado inestable determina la anchura del impulso de su salida.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Describir el funcionamiento básico de un monoestable. ■ Explicar cómo funciona un monoestable no redisparable. ■ Explicar cómo funciona un monoestable redisparable. ■ Configurar los monoestables 74121 y 74LS122 para obtener una anchura de impulso determinada. ■ Reconocer el símbolo de un *trigger* Schmitt y explicar qué significa.

▲ *Un monoestable genera un único impulso cada vez que se dispara.*

La Figura 7.44 muestra un monoestable (multivibrador de un solo estado) básico formado por una puerta lógica y un inversor. Cuando se aplica un impulso a la entrada de **disparo** (*trigger*), la salida de la puerta G_1 pasa a nivel BAJO. Esta transición de nivel ALTO a nivel BAJO se acopla por medio del condensador a la entrada del inversor G_2. La presencia de un aparente nivel BAJO en G_2 hace que su salida pase a nivel ALTO. Este nivel ALTO se realimenta a la puerta G_1, manteniendo su salida a nivel BAJO. Hasta este punto, el impulso de disparo ha hecho que la salida del monoestable, Q, sea un nivel ALTO.

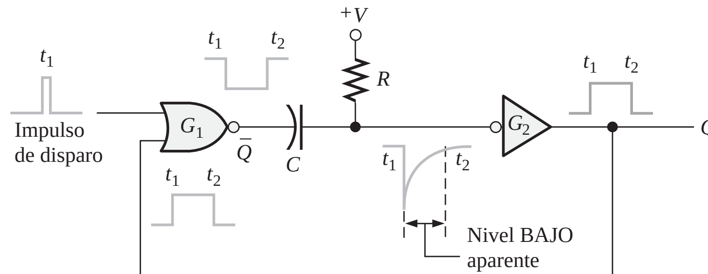


FIGURA 7.44 Circuito monoestable simple.

El condensador comienza inmediatamente a cargarse a través de R hasta alcanzar su tensión máxima. La velocidad de carga está determinada por la constante de tiempo RC. Cuando el condensador se carga hasta un determinado nivel, el cual aparece como un nivel ALTO en G_2, la salida pasa de nuevo a nivel BAJO.

En resumen, la salida del inversor G_2 pasa a nivel ALTO en respuesta a la entrada de disparo. Permanece a nivel ALTO durante un tiempo definido por la constante de tiempo, RC, y al final de este intervalo pasa a nivel BAJO. De esta manera, un único impulso estrecho produce un único impulso de salida cuyo período se controla mediante la constante de tiempo RC. Este modo de operación se puede ver en la Figura 7.44.

En la Figura 7.45(a) se muestra el típico símbolo lógico de un monoestable, y en la 7.45(b) se presenta el mismo símbolo con R y C externos. Los dos tipos fundamentales de circuitos integrados monoestables son los redisparables y los no redisparables.

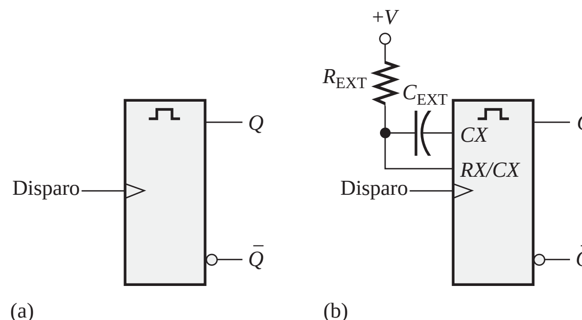


FIGURA 7.45 Símbolos lógicos básicos de los monoestables. *CX* y *RX* indican componentes externos.

Un monoestable no redisparable no responderá a ningún impulso de disparo adicional, desde el momento en que se pasa a su estado inestable hasta que retorna a su estado estable. En otras palabras, ignorará cualquier impulso de disparo que ocurra antes de que termine el periodo inestable. El tiempo que permanece el monoestable en su estado inestable es la anchura del impulso de salida.

La Figura 7.46 presenta un monoestable no redisparable, disparado a intervalos mayores y menores que su anchura de impulso. Observe que, en el segundo caso, los impulsos adicionales se ignoran.

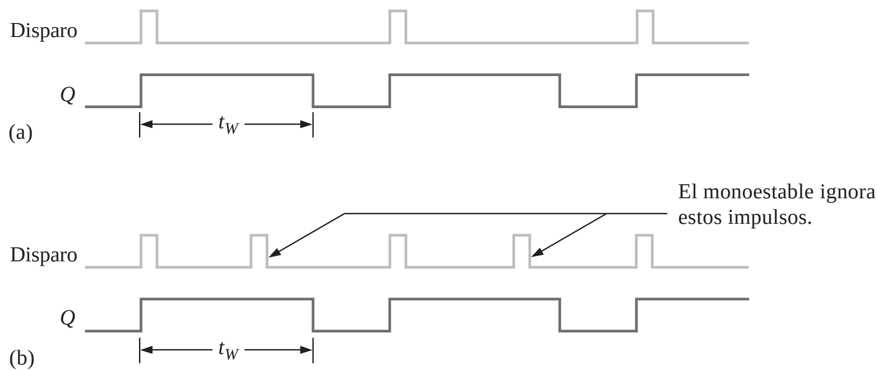


FIGURA 7.46 Funcionamiento de un monoestable no redisparable.

Un monoestable redisparable puede ser disparado antes de que retorne a su estado estable. El resultado del redisparo es una ampliación de la anchura del impulso, como se muestra en la Figura 7.47.

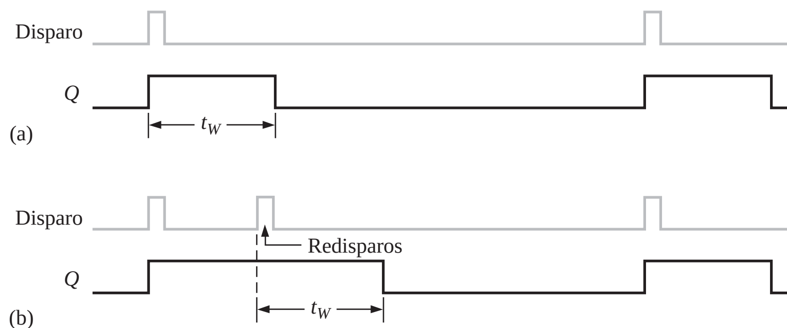
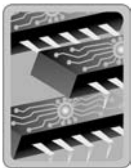


FIGURA 7.47 Funcionamiento de un monoestable redisparable.

MONOESTABLE NO REDISPARABLE 74121

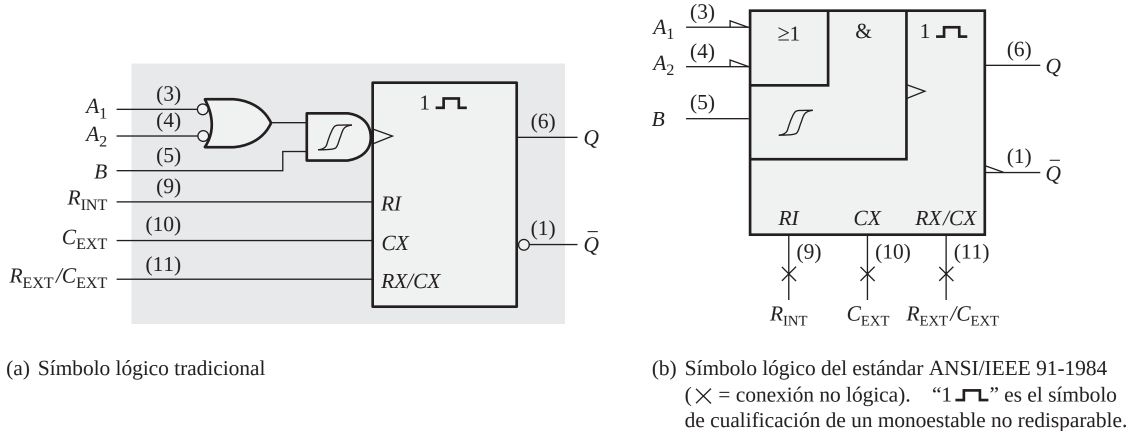


El 74121 es un ejemplo de un circuito monoestable integrado no redisparable. Como muestra la Figura 7.48, está previsto para conectarse a R y C externos. Las entradas etiquetadas como A_1, A_2 y B son entradas de activación de disparo. La entrada R_{INT} está conectada a una resistencia interna de temporización de 2 kΩ.

Establecimiento de la anchura del impulso. Cuando no se utiliza ningún componente de temporización externo y la resistencia de temporización interna (R_{INT}) se conecta a V_{CC}, como se muestra en la Figura 7.49(a) se produce un impulso típico de unos 30 ns de anchura. La anchura del impulso se puede ajustar entre 30 ns y 28 s utilizando los componentes externos. La Figura 7.49(b) muestra la conexión de una resistencia interna (2 kΩ) y un condensador externo. La parte (c) ilustra la conexión de una resistencia y un condensador externos. La anchura del impulso de salida se ajusta mediante los valores de la resistencia (R_{INT} = 2 kΩ, R_{EXT} variable) y del condensador de acuerdo con la siguiente fórmula:

Ecuación 7.1 $t_W = 0{,}7\ RC_{EXT}$

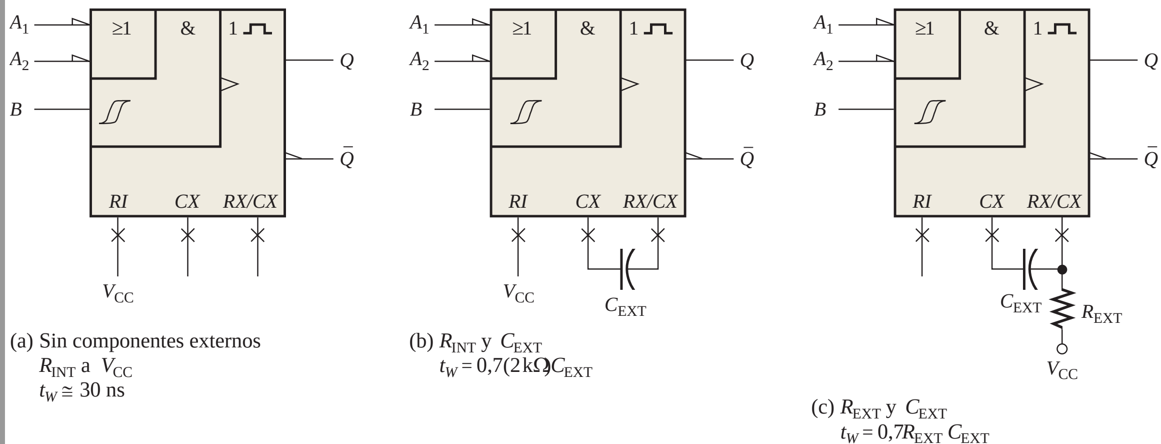
donde R puede ser tanto R_{INT} como R_{EXT}. Cuando R se expresa en kilohmios (kΩ) y C_{EXT} en picofaradios (pF), la anchura del impulso de salida t_W se obtiene en nanosegundos (ns).



(a) Símbolo lógico tradicional

(b) Símbolo lógico del estándar ANSI/IEEE 91-1984 (✕ = conexión no lógica). "1 ⎍" es el símbolo de cualificación de un monoestable no redisparable.

FIGURA 7.48 Símbolos lógicos del monoestable no redisparable 74121.



(a) Sin componentes externos
R_{INT} a V_{CC}
$t_W \cong 30$ ns

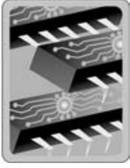
(b) R_{INT} y C_{EXT}
$t_W = 0{,}7(2\text{k}\Omega)C_{EXT}$

(c) R_{EXT} y C_{EXT}
$t_W = 0{,}7R_{EXT}\ C_{EXT}$

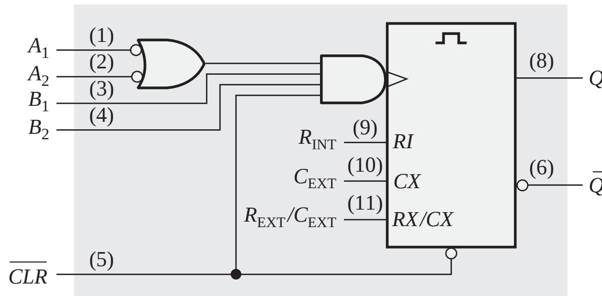
FIGURA 7.49 Tres maneras de ajustar la anchura de los impulsos en un 74121.

Símbolo del trigger Schmitt El símbolo ∫ indica una entrada de un *trigger* Schmitt. Este tipo de entrada emplea un circuito de umbral especial que produce **histéresis**, una característica que previene la conmutación errática entre estados cuando una tensión de disparo que varía muy lentamente se encuentra en las cercanías de un nivel de entrada crítica. Esto permite que se produzcan disparos fiables incluso cuando la entrada esté variando tan lentamente como a 1 voltio/segundo.

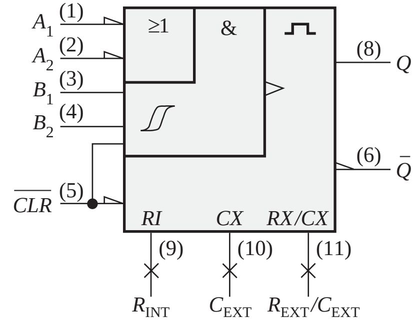
MONOESTABLE REDISPARABLE 74LS122



El 74LS122 es un ejemplo de un monoestable redisparable con entrada de borrado (*clear*). También está diseñado para añadir *R* y *C* externas, como muestra la Figura 7.50. Las entradas etiquetadas como A_1, A_2, B_1 y B_2 son entradas de activación de disparo.



(a) Símbolo lógico tradicional



(b) Símbolo lógico del ANSI/IEEE St. 91–1984 (× = conexión no lógica). ⎍ es el símbolo de cualificación de un monoestable disparable

FIGURA 7.50 Símbolo lógico del monoestable redisparable 74LS122.

Sin ningún componente adicional se obtiene un impulso de unos 45 ns de anchura. Se pueden conseguir impulsos más anchos mediante el uso de componentes externos. La fórmula general para calcular los valores de estos componentes para una determinada anchura del impulso (t_W) es:

Ecuación 7.2

$$t_W = 0,32RC_{\text{EXT}}\left(1+\frac{0,7}{R}\right)$$

donde 0,32 es una constante determinada por el tipo particular de monoestable, *R* se expresa en kΩ y puede ser tanto la resistencia interna como la externa, C_{EXT} se expresa en pF y t_W en ns. La resistencia interna vale 10 kΩ y puede utilizarse en lugar de una resistencia externa; observe la diferencia entre esta fórmula y la correspondiente del 74121, mostrada en la Ecuación 7.1.

EJEMPLO 7.12

Una determinada aplicación requiere un monoestable con una anchura de impulso de aproximadamente 100 ms. Utilizando un 74121, dibujar las conexiones y hallar los valores de sus componentes.

Solución

Arbitrariamente se selecciona R_{EXT} = **39 kΩ** y se calcula la capacidad necesaria.

$$t_W = 0,7R_{\text{EXT}}C_{\text{EXT}} \qquad C_{\text{EXT}} = \frac{t_W}{0,7R_{\text{EXT}}}$$

donde C_{EXT} está en pF, R_{EXT} está en kΩ y t_W en ns. Dado que 100 ms = 1×10^8 ns,

$$C_{\text{EXT}} = \frac{1\times10^8\,\text{ns}}{0,7(39\text{k}\Omega)} = 3,66\times10^{-6}\,\text{pF} = \mathbf{3,66\mu F}$$

Un condensador estándar de 3,3 μF proporcionará una anchura de impulso de salida de 91 ms. Las conexiones adecuadas se muestran en la Figura 7.51. Para conseguir una anchura de impulso más próxima a 100 ms, se pueden probar otras combinaciones de valores para R_{EXT} y C_{EXT}. Por ejemplo, R_{EXT} = 68 kΩ y C_{EXT} = 2,2 μF proporciona una anchura de impulso de 105 ms.

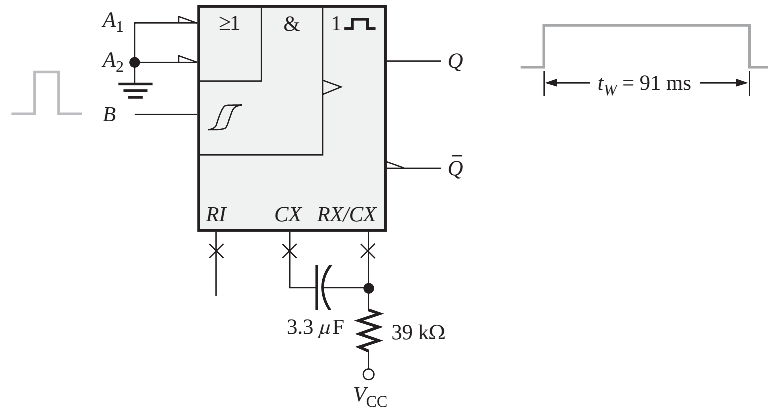


FIGURA 7.51

Problema relacionado Utilizar un condensador externo junto con R_{INT}, para producir un impulso de salida de anchura 10 μs en un 74121.

EJEMPLO 7.13

Determinar los valores de R_{EXT} y C_{EXT} que producen un impulso de anchura 1μs, cuando se conectan a un 74LS122.

Solución Suponer un valor de C_{EXT} = **560 pF** y luego calcular R_{EXT}. La anchura del impulso t_W debe expresarse en ns y C en pF. R_{EXT} se obtendrá en kΩ.

$$t_W = 0,32R_{\text{EXT}}C_{\text{EXT}}\left(1+\frac{0,7}{R_{\text{EXT}}}\right) = 0,32R_{\text{EXT}}C_{\text{EXT}} + 0,7\left(\frac{0,32R_{\text{EXT}}C_{\text{EXT}}}{R_{\text{EXT}}}\right)$$

$$= 0,32R_{\text{EXT}}C_{\text{EXT}} + (0,7)(0,32)C_{\text{EXT}}$$

$$R_{\text{EXT}} = \frac{t_W - (0,7)(0,32)C_{\text{EXT}}}{0,32C_{\text{EXT}}} = \frac{t_W}{0,32C_{\text{EXT}}} - 0,7$$

$$= \frac{100\,\text{ns}}{(0,32)560\text{ pF}} - 0,7 = \mathbf{4,88\,k\Omega}$$

	Utilizar un valor estándar de **4,7 kΩ**.
Problema relacionado	Dibujar las conexiones y calcular los valores de los distintos componentes para un monoestable 74LS122, que tiene un impulso de salida de 5 μs de anchura. Suponer que C_{EXT} = 560 pF.

Aplicación

Una aplicación práctica de los monoestables es un temporizador secuencial, que puede utilizarse para encender una serie de luces. Este tipo de circuito puede emplearse, por ejemplo, en una autopista en construcción para indicar cambios sucesivos de carril.

La Figura 7.52 muestra tres monoestables 74LS122 conectados como temporizador secuencial. Este circuito particular produce una secuencia de tres impulsos de 1 segundo. El primer monoestable se dispara mediante el cierre de un interruptor o mediante un impulso de entrada de baja frecuencia, que genera un impulso de salida de 1 segundo. Cuando el primer monoestable (ME 1) pasa a su estado estable y pasa a nivel BAJO, el segundo monoestable (ME 2) se dispara, produciendo también un impulso de salida de 1 segundo. Cuando este segundo impulso pasa a nivel BAJO, el tercer monoestable (ME 3) se dispara, produciéndose el tercer impulso de 1 segundo. El diagrama de tiempos de salida se muestra en la figura. Se pueden realizar variaciones de esta configuración básica para producir diversas salidas de temporización.

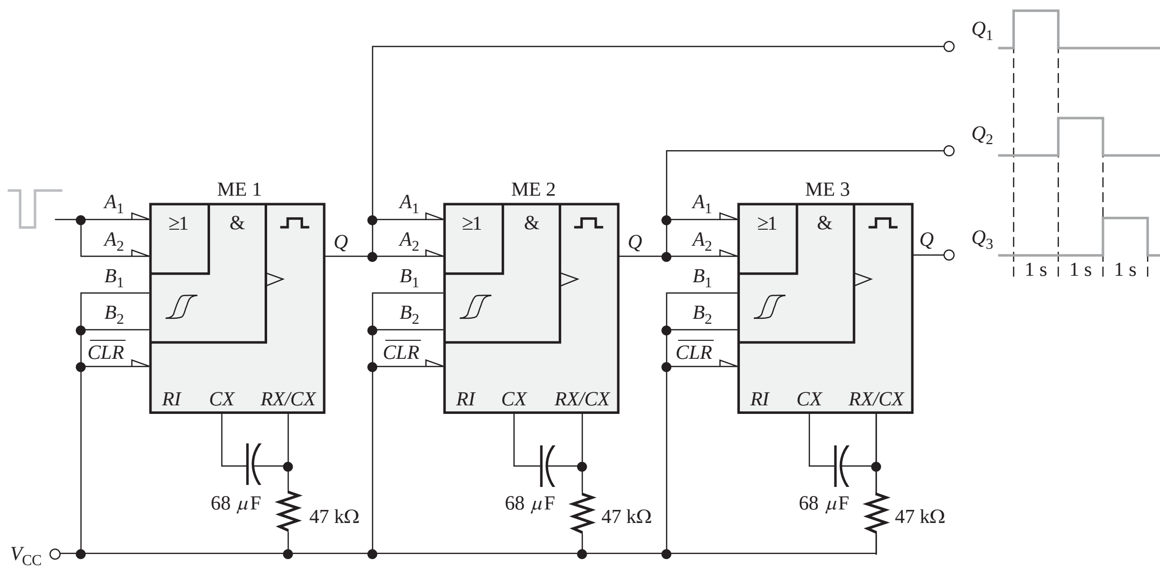


FIGURA 7.52 Circuito de temporización secuencial utilizando tres monoestables 74LS122.

REVISIÓN DE LA SECCIÓN 7.5

1. Describir las diferencias entre un monoestable no redisparable y uno redisparable.
2. ¿Cómo se ajusta la anchura del impulso en la mayoría de los circuitos integrados monoestables?

7.6 EL TEMPORIZADOR 555

El ***temporizador*** 555 es un dispositivo versátil y muy utilizado, porque puede ser configurado de dos modos distintos, bien como multivibrador monoestable o como multivibrador aestable (oscilador). Un multivibrador aestable no tiene estados estables y varía, por tanto, una y otra vez (oscila) entre dos estados inestables, sin utilizar un circuito de disparo externo.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Describir los elementos básicos de un temporizador 555. ■ Configurar un temporizador 555 como monoestable. ■ Configurar un temporizador 555 como oscilador.

Funcionamiento básico

▲ *Un temporizador 555 puede operar como monoestable o como oscilador (aestable).*

En la Figura 7.53 se muestra un diagrama funcional con los componentes internos de un temporizador 555. Los comparadores son dispositivos cuyas salidas están a nivel ALTO cuando la tensión en la entrada positiva (+) es mayor que la tensión en la entrada negativa (-), y están a nivel BAJO cuando la tensión de entrada negativa es mayor que la tensión de entrada positiva. El divisor de tensión, formado por tres resistencias de 5kΩ, proporciona un nivel de disparo de $1/3V_{CC}$ y un nivel umbral de $2/3V_{CC}$. La entrada de la tensión de control (pin 5) se puede emplear para ajustar externamente los niveles de disparo y umbral a otros valores en caso necesario. Cuando la entrada de disparo, normalmente a nivel ALTO, desciende momentáneamente por debajo de $1/3\ V_{CC}$, la salida del comparador B conmuta de nivel BAJO a nivel ALTO y pone en estado SET al latch S-R, haciendo que la salida (pin 3) pase a nivel ALTO y bloqueando el transistor de descarga Q_1. La salida permanecerá a nivel ALTO hasta que la tensión umbral, normal-

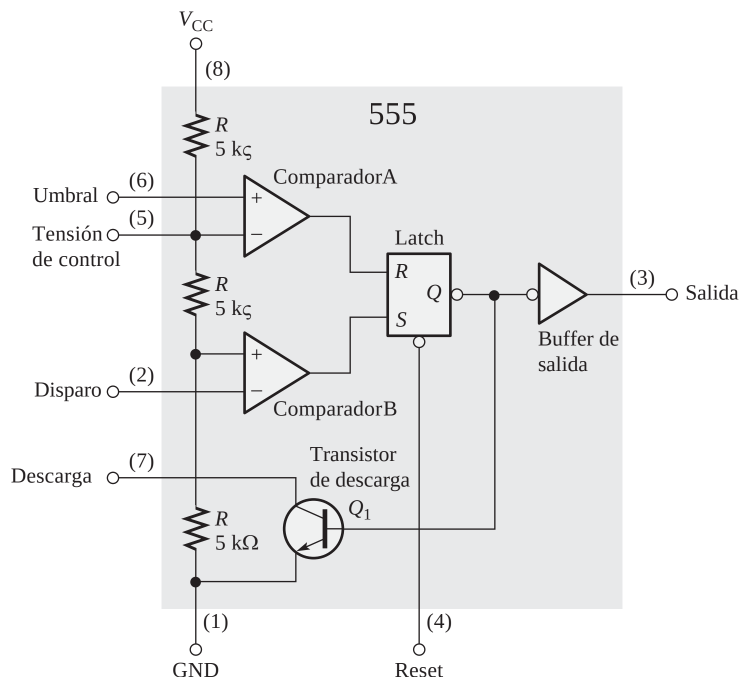


FIGURA 7.53 Diagrama funcional interno de un temporizador 555 (la numeración de pines se indica entre paréntesis).

mente a nivel BAJO sobrepase 2/3 *de* V_{CC} y haga que la salida del comparador A conmute de nivel BAJO a nivel ALTO. Esto hace que el latch pase a estado RESET, con lo que la salida se pone de nuevo a nivel BAJO, de manera que el transistor de descarga se activa. La entrada de puesta a cero (RESET) externa se puede utilizar para poner el latch a cero, independientemente del circuito umbral. Las entradas de disparo y umbral (pines 2 y 6) se controlan mediante componentes externos, para establecer el modo de funcionamiento como monoestable o aestable.

Funcionamiento como monoestable

Para configurar un temporizador 555 como monoestable no redisparable, se utilizan una resistencia y un condensador externos, tal como se muestra en la Figura 7.54. La anchura del impulso de salida se determina mediante la constante de tiempo, que se calcula a partir de R_1 y C_1 según la siguiente fórmula:

Ecuación 7.3 $t_W = 1{,}1\ R_1C_1$

La entrada de la tensión de control no se utiliza y se conecta a un condensador de desacoplo C_2, para evitar la aparición de ruido que pudiera afectar los niveles umbral y de disparo

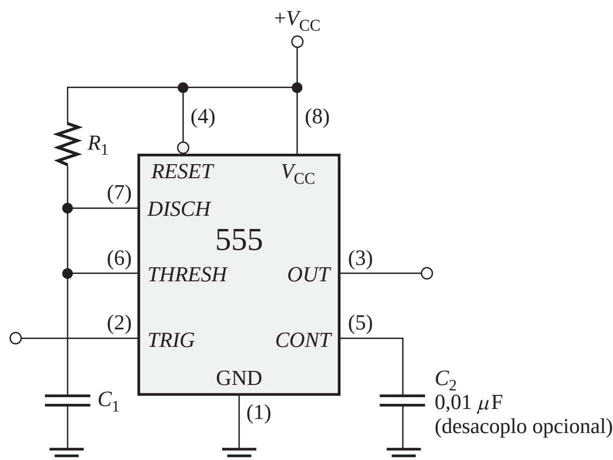


FIGURA 7.54 El temporizador 555 conectado como monoestable.

Antes de aplicar el impulso de disparo, la salida está a nivel BAJO y el transistor de descarga Q_1 *conduce*, manteniendo C_1 descargado, como se muestra en la Figura 7.55(a). Cuando se aplica un impulso de disparo negativo en el instante t_0, la salida pasa a nivel ALTO y el transistor de descarga se *bloquea*, permitiendo al condensador C_1 comenzar a cargarse a través de R_1, como se muestra en la parte (b). Cuando C_1 se ha cargado hasta 1/3 *de* V_{CC}, la salida pasa de nuevo a nivel BAJO en t_1 y Q_1 entra en conducción inmediatamente, descargándose C_1, como se indica en la parte (c). Como puede ver, la velocidad de carga de C_1 determina cuánto tiempo va a estar la salida a nivel ALTO.

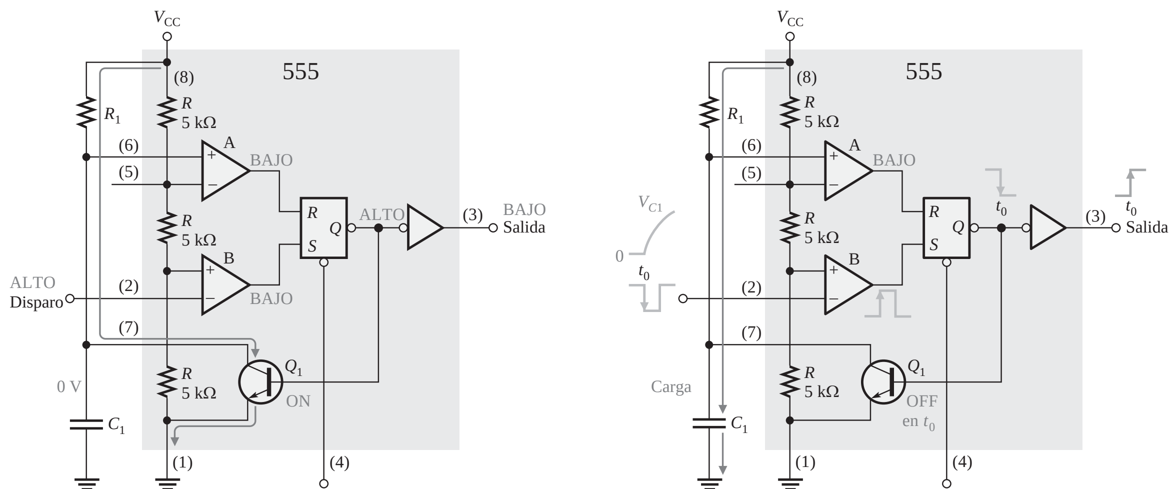
EJEMPLO 7.14

¿Cuál es la anchura del impulso de salida para un circuito monoestable 555 con R_1 = 2,2 kΩ y C_1 = 0,01 μF?

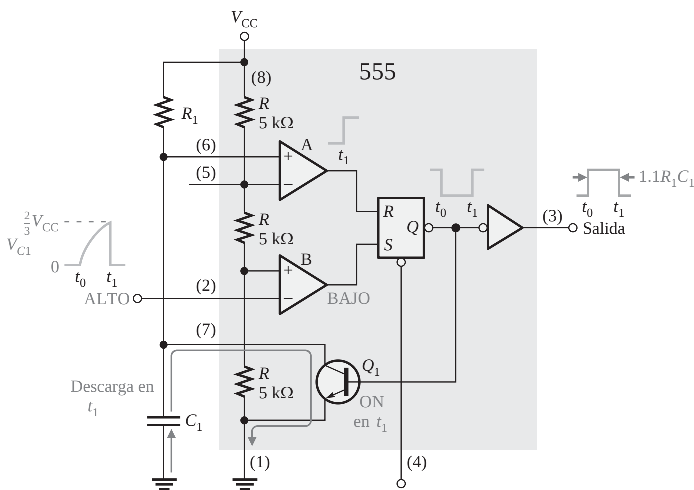
Solución A partir de la Ecuación 7.3, tenemos que la anchura del impulso es:

$$t_W = 1{,}1R_1C_1 = 1{,}1(2{,}2\ \text{k}\Omega)(0{,}01\mu\text{F}) = \mathbf{24{,}2\mu s}$$

Problemas relacionado Para C_1 = 0,01μF, determinar el valor de R_1 para una anchura de impulso de 1ms.



(a) Antes del disparo. (El camino de la corriente lo indica la flecha.) (b) Durante el disparo



(c) Al final del intervalo de carga

FIGURA 7.55 Funcionamiento del temporizador 555 configurado como monoestable.

Funcionamiento como aestable

En la Figura 7.56 se muestra un temporizador 555 conectado para funcionar como multivibrador ***aestable***, que es un **oscilador** no sinusoidal. Observe que, en este caso, la entrada umbral (*THRESH*) está conectada a la entrada de disparo (*TRIG*). Los componentes externos R_1, R_2 y C_1 conforman la red de temporización que

determina la frecuencia de oscilación. El condensador C_2 de 0,01 μF conectado a la entrada de control (*CONT*) sirve únicamente para desacoplar y no afecta en absoluto al funcionamiento del resto del circuito; en algunos casos se puede eliminar.

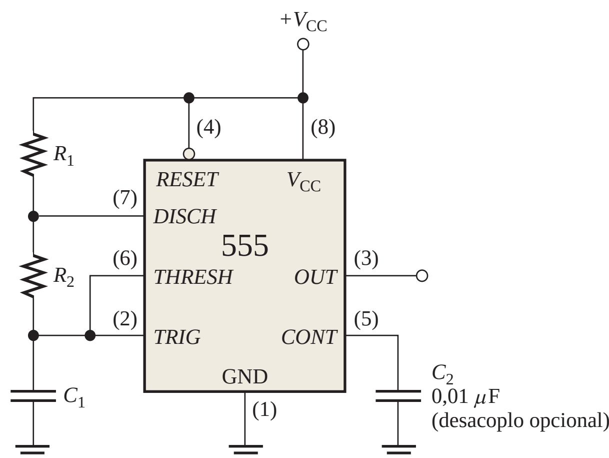


FIGURA 7.56 El temporizador 555 configurado como multivibrador aestable (oscilador).



NOTAS INFORMÁTICAS

Todas las computadoras requieren una fuente de temporización para proporcionar señales de reloj precisas. La sección de temporización controla todas las temporizaciones del sistema y es responsable del correcto funcionamiento del hardware del sistema. Normalmente, la sección de temporización consta de un oscilador controlado de cristal y de contadores para realizar la división de frecuencia. El uso de un oscilador de alta frecuencia y de divisores para obtener una frecuencia menor proporciona una mayor precisión y estabilidad en frecuencia.

Inicialmente, cuando se conecta la alimentación, el condensador (C_1) está descargado y, por tanto, la tensión de disparo (pin 2) es 0 V. Esto da lugar a que la salida del comparador B esté a nivel ALTO y la salida del comparador A a nivel BAJO, forzando la salida del latch, y por consiguiente la base de Q_1 a nivel BAJO, manteniendo el transistor bloqueado. A continuación, C_1 comienza a cargarse a través de R_1 y R_2, tal como se indica en la Figura 7.57. Cuando la tensión del condensador alcanza el valor de 1/3 V_{CC}, el comparador B cambia a su nivel de salida BAJO, y cuando la tensión del condensador alcanza el valor de 2/3 V_{CC}, el comparador A cambia a su nivel de salida ALTO. Esto pone en estado de RESET al latch, haciendo que la base de Q_1 pase a nivel ALTO, activando el transistor. Esta secuencia origina un camino de descarga para el condensador a través de R_2 y del transistor, tal como se indica. El condensador comienza ahora a descargarse, haciendo que el comparador A pase a nivel BAJO. En el momento en que el condensador se descarga hasta el valor 1/3 V_{CC}, el comparador B conmuta a nivel ALTO, poniendo al latch en estado SET, lo que hace que la base de Q_1 se ponga a nivel BAJO, bloqueando el transistor. De nuevo comienza otro ciclo de carga, y el proceso completo se repite. El resultado es una señal de salida rectangular cuyo ciclo de trabajo depende de los valores de R_1 y R_2. La frecuencia de oscilación viene dada por la siguiente fórmula, o puede también hallarse utilizando el gráfico de la Figura 7.58.

Ecuación 7.4

$$f = \frac{1,44}{(R_1 + 2R_2)C_1}$$

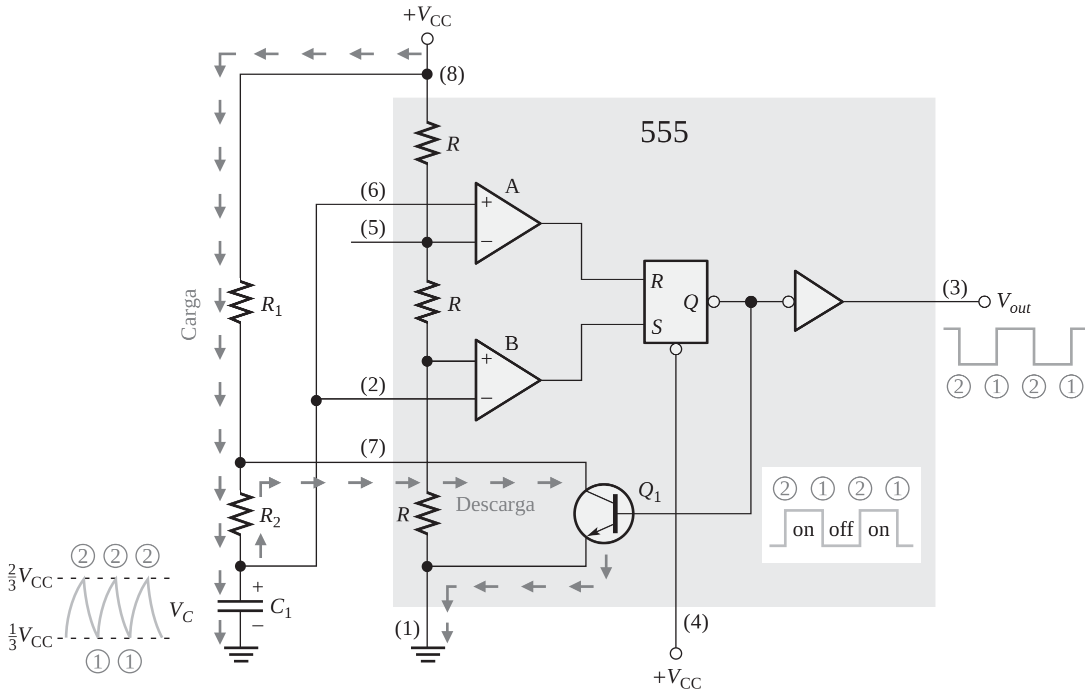


FIGURA 7.57 Funcionamiento del temporizador 555 configurado en modo aestable.

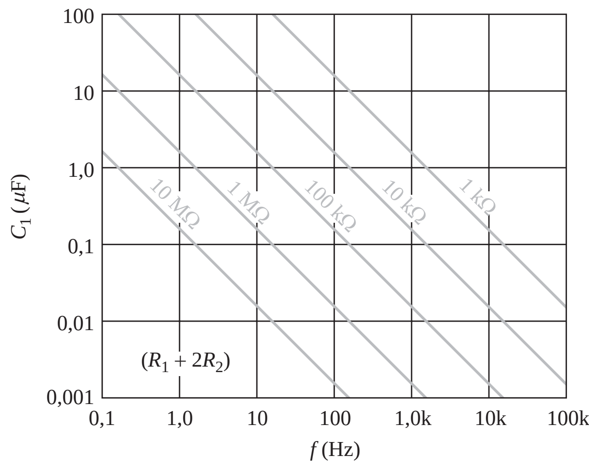


FIGURA 7.58 Frecuencia de oscilación en función de C_1 y $R_1 + 2R_2$. Las líneas diagonales representan los valores de $R_1 + 2R_2$.

El ciclo de trabajo de la salida puede ser ajustado seleccionando R_1 y R_2. Dado que C_1 se carga a través de $R_1 + R_2$ y se descarga únicamente a través R_2, se pueden conseguir ciclos de trabajo de un mínimo del 50 por ciento aproximadamente, si $R_2 >> R_1$, de forma que los tiempos de carga y descarga sean aproximadamente iguales.

La expresión para el ciclo de trabajo se obtiene de la manera siguiente. El intervalo de tiempo en que la salida está a nivel ALTO (t_H) representa lo que tarda C_1 en cargarse desde 1/3 V_{CC} hasta 2/3 V_{CC}. Esto se expresa como:

Ecuación 7.5 $t_H = 0{,}7(R_1 + R_2)C_1$

El intervalo de tiempo durante el que la salida está a nivel BAJO (t_L) representa lo que tarda C_1 en descargarse desde 1/3V_{CC} hasta 2/3V_{CC}. Esto se expresa como:

Ecuación 7.6 $t_L = 0{,}7R_2C_1$

El período, T, de la señal de salida es la suma de t_H y t_L. Esto es el recíproco de f en la Ecuación (7.4).

$$T = t_H + t_L = 0{,}7(R_1 + 2R_2)C_1$$

Finalmente, el ciclo de trabajo es:

$$\text{Ciclo de trabajo} = \frac{t_H}{T} = \frac{t_H}{t_H + t_L}$$

Ecuación 7.7
$$\text{Ciclo de trabajo} = \left(\frac{R_1 + R_2}{R_1 + 2R_2}\right)100\%$$

Para conseguir ciclos de trabajo menores que el 50 por ciento, se puede modificar el circuito de la Figura 7.56, de modo que C_1 se cargue sólo a través de R_1 y se descargue a través de R_2. Esto se consigue mediante un diodo D_1 colocado tal y como se muestra en la Figura 7.59. El ciclo de trabajo se puede hacer menor que el 50 por ciento, haciendo R_1 menor que R_2. Bajo esta condición, la expresión para el ciclo de trabajo es:

Ecuación 7.8
$$\text{Ciclo de trabajo} = \left(\frac{R_1}{R_1 + R_2}\right)100\%$$

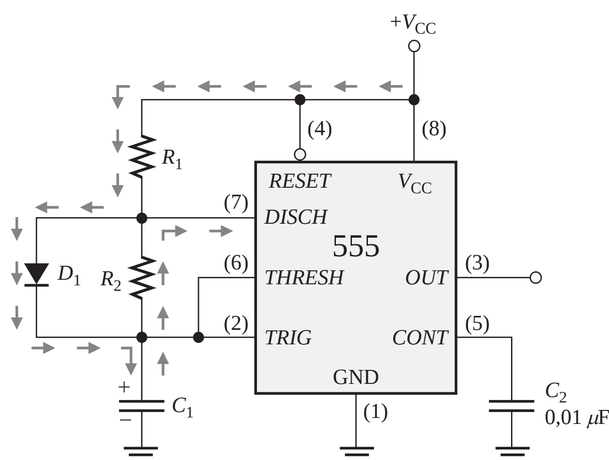


FIGURA 7.59 La adición de un diodo D_1 permite ajustar el ciclo de trabajo de la salida a un valor menor del 50 por ciento, haciendo $R_1 < R_2$.

EJEMPLO 7.15

En la Figura 7.60 se muestra un temporizador 555 configurado para funcionar en modo aestable (oscilador). Determinar la frecuencia de la salida y el ciclo de trabajo.

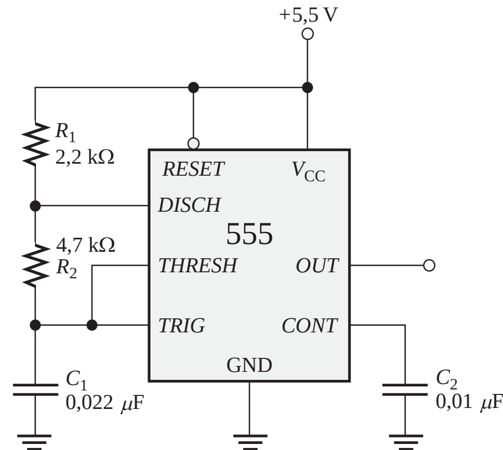


FIGURA 7.60

Solución Utilizar las ecuaciones 7.4 y 7.7.

$$f = \frac{1{,}44}{(R_1 + 2R_2)C_1} = \frac{1{,}44}{(2{,}2\text{k}\Omega + 9{,}4\text{k}\Omega)0{,}22\mu\text{F}} = \mathbf{5{,}64\,kHz}$$

$$\text{Ciclo de trabajo} = \left(\frac{R_1 + R_2}{R_1 + 2R_2}\right)100\% = \left(\frac{2{,}2\text{k}\Omega + 4{,}7\text{k}\Omega}{2{,}2\text{k}\Omega + 9{,}4\text{k}\Omega}\right)100\% = \mathbf{59{,}5\%}$$

Problema relacionado Determinar el ciclo de trabajo en la Figura 7.60 si se conecta un diodo en paralelo con R_2, como se indica en la Figura 7.59.

REVISIÓN DE LA SECCIÓN 7.6

1. Explicar la diferencia de funcionamiento entre un multivibrador aestable y un multivibrador monoestable.
2. Para un determinado multivibrador aestable, t_H = 15 ms y T = 20 ms. ¿Cuál es el ciclo de trabajo de la salida?

7.7 LOCALIZACIÓN DE AVERÍAS

En la industria, es una práctica habitual probar los nuevos diseños de circuitos para asegurarse de que funcionan como se ha especificado. Usualmente, se realiza un montaje provisional de los nuevos diseños de función fija y se prueban antes de que finalice el diseño. El término *montaje provisional* se refiere a la forma en que el nuevo circuito se ensambla para poder verificar su funcionamiento y, en caso de encontrarse fallos, eliminarlos antes de que se construya el prototipo.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Describir cómo la temporización de un circuito puede producir *glitches*. ■ Afrontar la depuración de un nuevo diseño con una mayor comprensión y conciencia de los problemas potenciales.

El circuito que se muestra en la Figura 7.61(a) genera dos señales de reloj (CLK A y CLK B), siendo la aparición de los impulsos alternativa. Cada señal tiene la frecuencia mitad de la señal de reloj original (CLK), como se muestra en el diagrama de tiempos ideal de la parte (b).

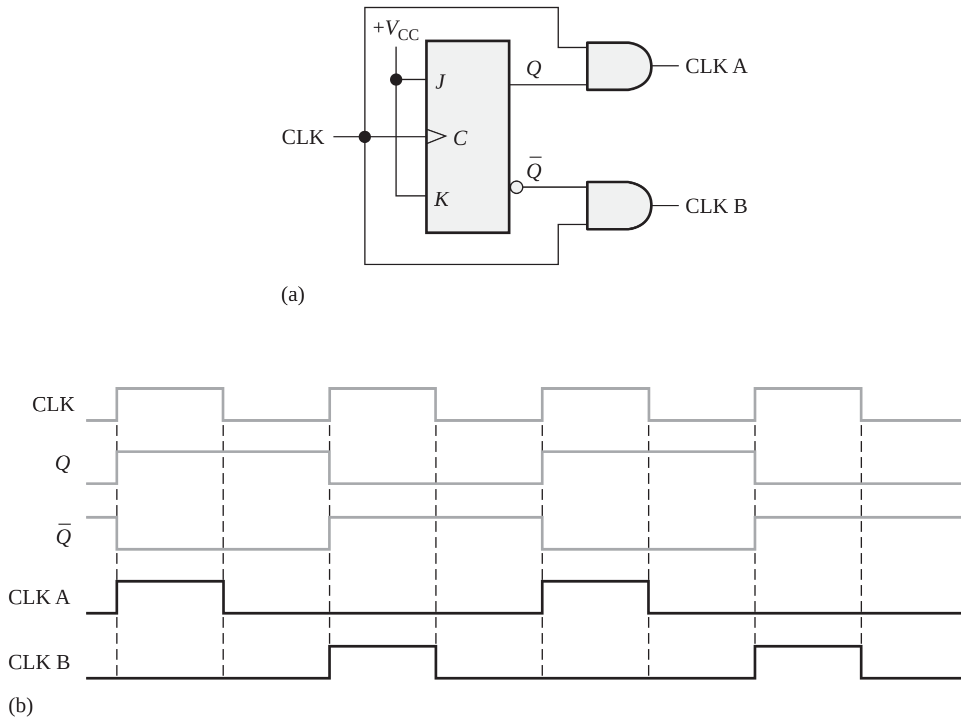
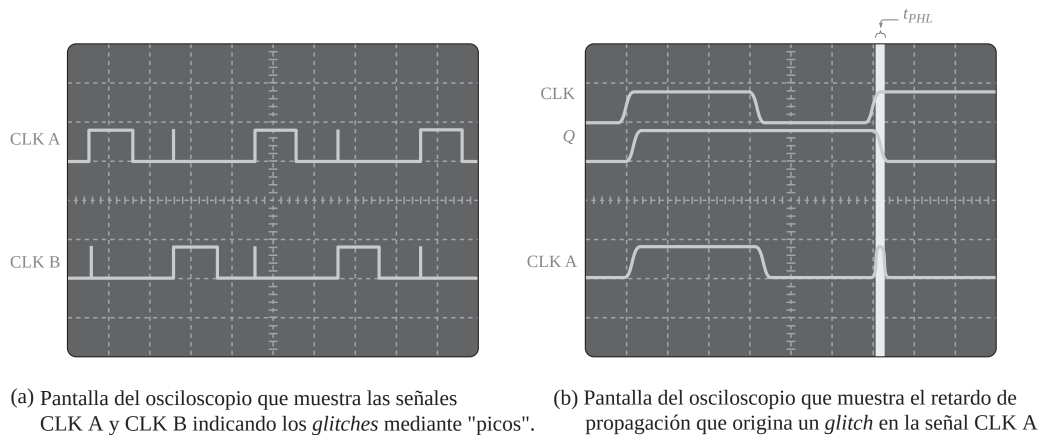


FIGURA 7.61 Generador de reloj de dos fases con señales ideales.



(a) Pantalla del osciloscopio que muestra las señales CLK A y CLK B indicando los *glitches* mediante "picos".

(b) Pantalla del osciloscopio que muestra el retardo de propagación que origina un *glitch* en la señal CLK A

FIGURA 7.62 Pantallas del osciloscopio para el circuito de la Figura 7.61.

Cuando se prueba el circuito, las señales CLK A y CLK B se presentan en el osciloscopio o en el analizador lógico como se muestra en la Figura 7.62(a). Puesto que se observan *glitches* en ambas señales, deducimos que existe algún fallo en el circuito, bien en su diseño básico o en la manera en que está conectado. Investigaciones posteriores revelan que los *glitches* son causados por una condición de **concurrencia** entre la señal CLK y las señales Q y $\bar{Q}$ en las entradas de las puertas AND. Como se muestra en la Figura 7.62(b), los retardos de propagación entre CLK y Q y $\bar{Q}$ dan lugar a una coincidencia de corta duración de niveles ALTOS en los flancos anteriores de los impulsos alternos de reloj. Por tanto, existe un error de diseño.

El problema puede ser corregido mediante el uso de un flip-flop disparado por flanco negativo en lugar del dispositivo disparado por flanco positivo, como se observa en la Figura 7.63(a). Aunque los retardos de propagación entre CLK y Q y $\bar{Q}$ siguen existiendo, éstos se inician en los flancos posteriores del reloj (CLK), eliminando por tanto los *glitches*, como se muestra en el diagrama de tiempos de la Figura 7.63(b).

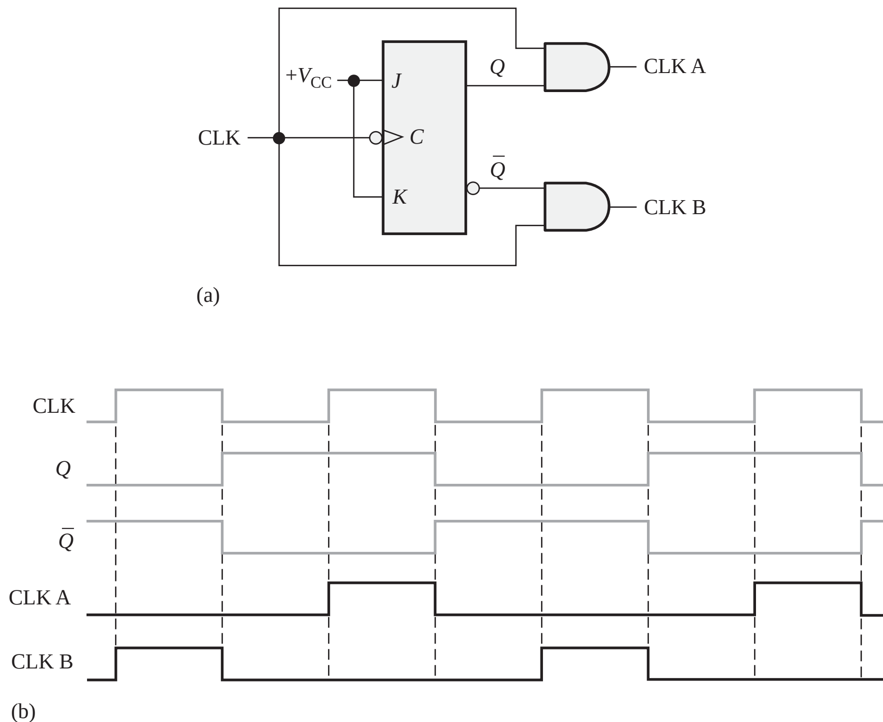


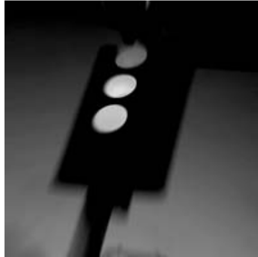
FIGURA 7.63 Generador de reloj de dos fases, que utiliza flip-flops disparados por flanco negativo para eliminar los glitches.

CONSEJOS PRÁCTICOS

Los *glitches* que se producen en los sistemas digitales son muy rápidos (y extremadamente cortos en duración) y pueden ser difíciles de ver en un osciloscopio, especialmente a muy bajas velocidades de barrido. Sin embargo, un analizador lógico puede mostrar un glitch fácilmente. Para localizar *glitches* utilizando un analizador lógico, seleccione el modo "*latch*" o el muestreo transicional, si está disponible. En el modo *latch*, el analizador busca un cambio de nivel de tensión. Cuando se produce un cambio, incluso aunque sea de una duración extremadamente corta (unos pocos nanosegundos), la información se almacena en la memoria del analizador como cualquier otro dato muestreado. Cuando se muestran los datos, el *glitch* se presentará como un cambio obvio en los datos muestreados, por lo que se puede identificar fácilmente.

REVISIÓN DE LA SECCIÓN 7.7

1. ¿Se puede usar un flip-flop D disparado por flanco negativo en el circuito de la Figura 7.63?
2. ¿Qué dispositivo puede utilizarse como reloj en el circuito de la Figura 7.63?



APLICACIÓN A LOS SISTEMAS DIGITALES

En esta aplicación a los sistemas digitales, vamos a continuar trabajando con el sistema de control de semáforos que se inició en el Capítulo 6. En el capítulo anterior se ha desarrollado la lógica combinacional.

En este capítulo, se van a desarrollar los circuitos de temporización, que generan los intervalos de tiempo de 4 s para la luz ámbar y de 25 s para las luces roja y verde, así como una señal de reloj de 4 Hz. El diagrama de bloques global del sistema de control de los semáforos que se presentó en el Capítulo 6 se incluye aquí de nuevo, en la Figura 7.64, como referencia.

Requisitos generales de los circuitos de temporización

Los circuitos de temporización constan de tres partes: el temporizador de 4 s, el temporizador de 25 s y el oscilador de 10 kHz, como se muestra en el diagrama de bloques de la Figura 7.65. Los temporizadores de 4 s y 25 s se implementan mediante monoestables 74121, como muestran las Figura 7.66 (a) y (b). El oscilador de 10 kHz se implementa con un temporizador 555 como se muestra en la Figura 7.66(c).

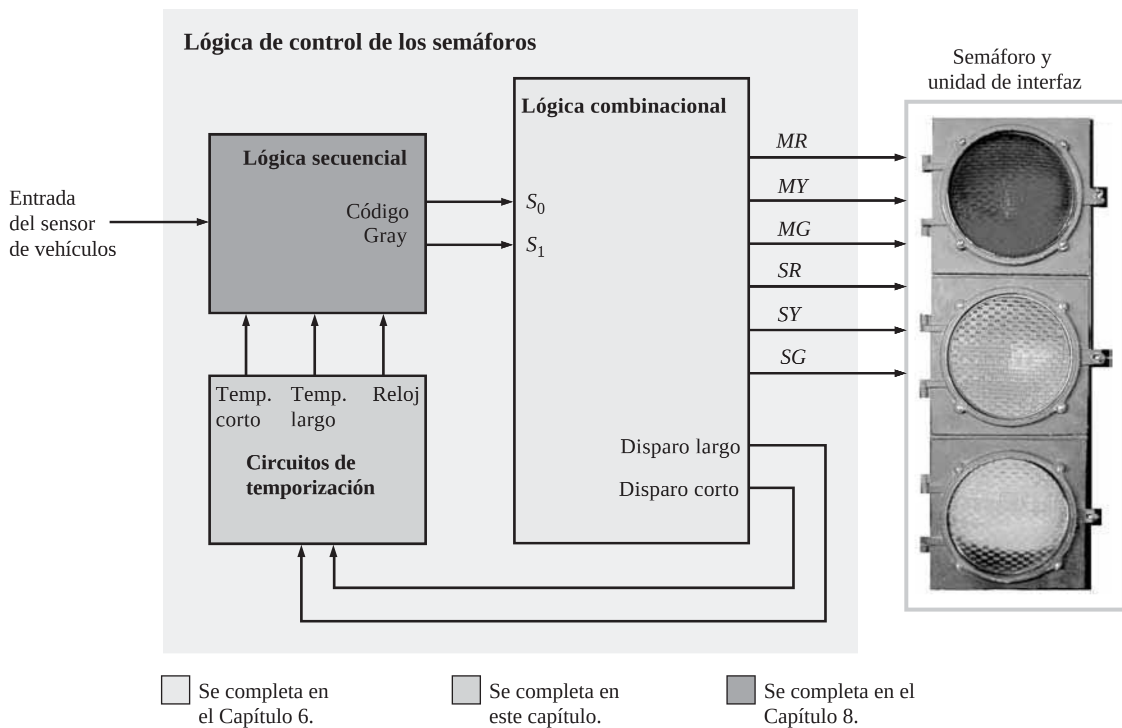


FIGURA 7.64 Diagrama de bloques del sistema de control de semáforos.

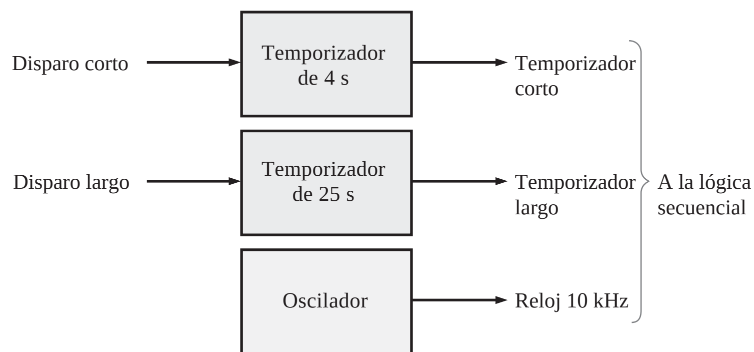
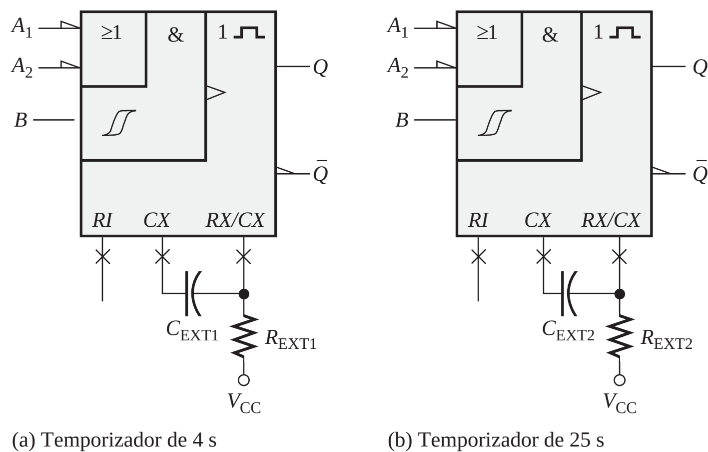


FIGURA 7.65 Diagrama de bloques de los circuitos de temporización.



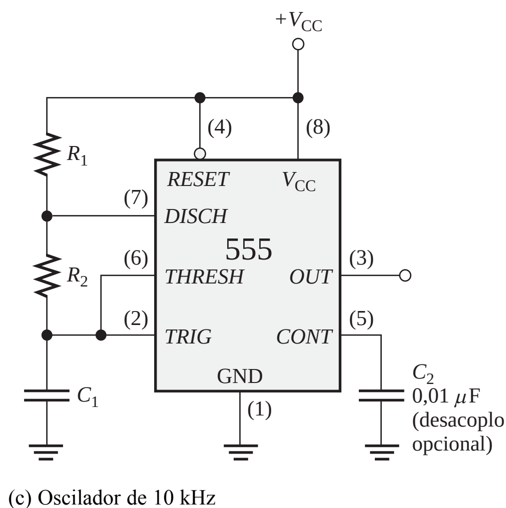


FIGURA 7.66 Circuitos de temporización.

Práctica de sistemas

- ***Actividad 1.*** Determinar los valores de los componentes *R* y *C* externos para el temporizador de 4 s de la Figura 7.66(a).
- ***Actividad 2.*** Determinar los valores de los componentes *R* y *C* externos para el temporizador de 25 s de la Figura 7.66(b).
- ***Actividad 3.*** Determinar los valores de los componentes *R* y *C* externos para el oscilador 555 de 10 kHz de la Figura 7.66(c).

RESUMEN

- Los símbolos de los *latches* y flip-flops se muestran en la Figura 7.67.

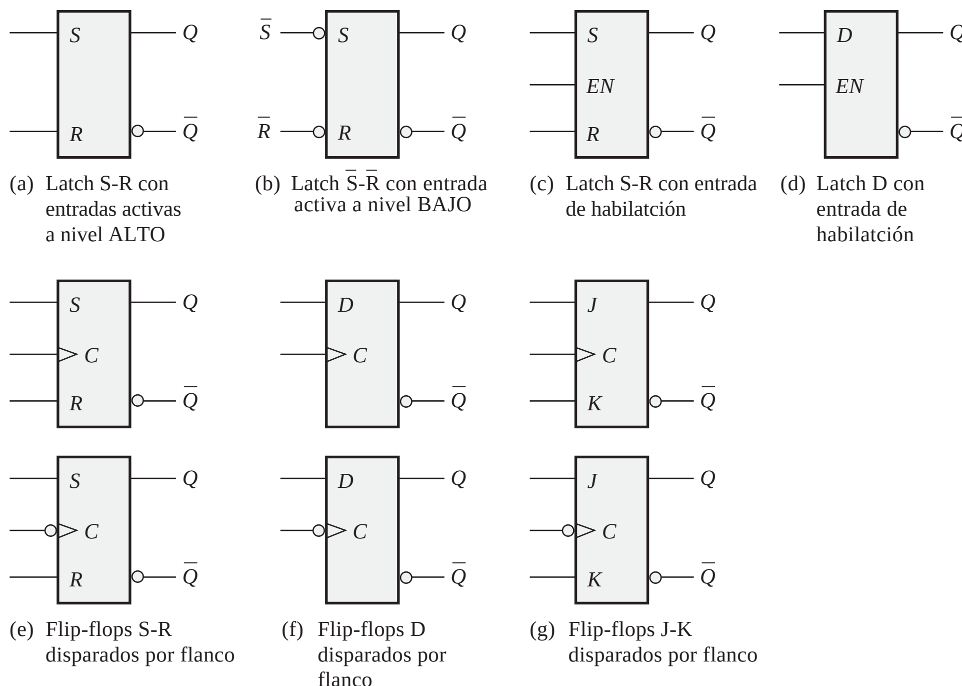


FIGURA 7.67

- Los *latches* son dispositivos biestables cuyo estado depende normalmente de entradas asíncronas.
- Los flip-flops disparados por flanco son dispositivos de dos estados con entradas síncronas, cuyo estado depende de las entradas sólo durante la transición de disparo de un impulso de reloj. Los cambios en las salidas ocurren durante las transiciones de disparo del reloj.
- Los multivibradores monoestables tienen un único estado estable. Cuando se dispara un monoestable, la salida pasa a su estado inestable durante un tiempo que está determinado por un circuito *RC*.
- Los multivibradores aestables no poseen ningún estado estable y se utilizan como osciladores para generar señales de temporización en los sistemas digitales.

PALABRAS CLAVE

Las palabras clave y otros términos que se han resaltado en negrita se encuentran en el glosario final del libro.

Aestable Que no tiene ningún estado estable. Un multivibrador aestable oscila entre dos estados semi-estables.

Basculación Acción de un flip-flop cuando cambia de estado con cada impulso de reloj.

Biestable Que tiene dos estados estables. Los flip-flops y los latches son multivibradores biestables.

Borrado (*clear*) Entrada asíncrona utilizada para resetear un flip-flop (hacer que la salida Q sea 0).

Disipación de potencia La cantidad de potencia requerida por un circuito.

Flip-flop D Un tipo de multivibrador biestable en el que la salida sigue al estado de la entrada D en el flanco de disparo de un impulso de reloj.

Flip-flop disparado por flanco Un tipo de flip-flop en el que los datos se introducen y aparecen en la salida durante el mismo flanco del impulso del reloj.

Flip-flop J-K Un tipo de flip-flop que puede funcionar en los modos de SET, RESET, no cambio y basculación.

Inicialización (*preset*) Entrada asíncrona para inicializar un flip-flop.

Latch Dispositivo digital biestable utilizado para almacenar un bit.

Monoestable Que tiene un solo estado estable. Un multivibrador monoestable, o sencillamente un monoestable, produce un único impulso en respuesta a una entrada de disparo.

Reloj La entrada de disparo de un flip-flop.

Retardo de propagación El intervalo de tiempo requerido después de haberse aplicado una señal de entrada para que se produzca un cambio en la salida.

RESET Estado de un flip-flop o *latch* cuando la salida es 0. La acción de producir un estado de desactivación.

SET Estado de un flip-flop o latch cuando la salida es 1. La acción de producir un estado de activación.

Síncrono Que tiene una relación temporal fija.

Temporizador Circuito que puede ser utilizado como monoestable o como oscilador.

Tiempo de establecimiento (*set-up*) Intervalo de tiempo que los niveles de control deben mantenerse en las entradas de un circuito digital, como puede ser un flip-flop, antes del flanco de disparo del impulso de reloj.

Tiempo de mantenimiento (*hold*) El intervalo de tiempo que los niveles de control deben permanecer en las entradas de un flip-flop después del flanco de disparo del reloj, de manera que se active fiablemente el dispositivo.

AUTOTEST

Las respuestas se encuentran al final del capítulo.

1. Si un *latch* S-R tiene un 1 en la entrada S y un 0 en la entrada R y a continuación la entrada S pasa a 0, el *latch* estará en

(a) estado SET **(b)** estado RESET

(c) condición no válida **(d)** borrado

2. El estado no válido de un *latch* S-R se produce cuando

(a) $S = 1, R = 0$ **(b)** $S = 0, R = 1$

(c) $S = 1, R = 1$ **(d)** $S = 0, R = 0$

3. En un *latch* D con entrada de habilitación, la salida Q siempre es igual a la entrada D

(a) antes del impulso de habilitación

(b) durante el impulso de habilitación

(c) inmediatamente después del impulso de habilitación

(d) respuestas (b) y (c)

4. Al igual que el *latch*, el flip-flop pertenece a una categoría de circuitos lógicos conocidos como:

(a) multivibradores monoestables

(b) multivibradores biestables

(c) multivibradores aestables

(d) monoestables

5. El propósito de la entrada de reloj en un flip-flop es:

(a) borrar el dispositivo

(b) activar (SET) el dispositivo

(c) obligar siempre a la salida a cambiar de estado

(d) obligar a la salida a asumir un estado dependiente de las entradas de control (S-R, J-K o D)

6. En un flip-flop D disparado por flanco,

(a) un cambio en el estado del flip-flop puede producirse sólo en un flanco del impulso de reloj.

(b) el estado al que pasa el flip-flop depende de la entrada D

(c) la salida sigue a la entrada en cada impulso de reloj

(d) todas las respuestas

7. Una característica que diferencia al flip-flop J-K del flip-flop S-R es

(a) la condición de basculación

(b) la entrada de inicialización

(c) el tipo de reloj

(d) la entrada de borrado

8. Un flip-flop está en la condición de basculación cuando

(a) $J = 1, K = 0$

(b) $J = 1, K = 1$

(c) $J = 0, K = 0$

(d) $J = 0, K = 1$

9. Un flip-flop J-K con $J = 1$ y $K = 1$ tiene una entrada de reloj de 10 kHz. La salida Q es:

(a) constantemente un nivel ALTO

(b) constantemente un nivel BAJO

(c) una onda cuadrada de 10 kHz

(d) una onda cuadrada de 5 kHz

10. Un monoestable es un tipo de:

(a) multivibrador monoestable

(b) multivibrador aestable

(c) temporizador

(d) las respuestas (a) y (c)

(e) las respuestas (b) y (c)

11. La anchura del impulso de salida de un monoestable no disparable depende de:

(a) los intervalos de disparo

(b) la tensión de alimentación

(c) una resistencia y un condensador

(d) la tensión umbral

12. Un multivibrador aestable:

(a) requiere una entrada de disparo periódica

(b) no tiene ningún estado estable

(c) es un oscilador

(d) produce un impulso de salida periódico

(e) las respuestas (a), (b), (c) y (d)

(f) las respuestas (b), (c) y (d)

PROBLEMAS

Las respuestas a los problemas impares se encuentran al final del libro.

SECCIÓN 7.1 *Latches*

1. Si se aplican las señales de la Figura 7.68 a un *latch* S-R con entradas activas a nivel BAJO, dibujar la forma de onda de salida Q resultante en función de las entradas. Suponer que, inicialmente, Q está a nivel BAJO.

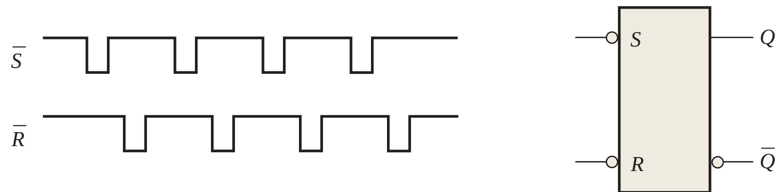


FIGURA 7.68

2. Resolver el Problema 1 para las formas de onda de entrada de la Figura 7.69, aplicadas a un *latch* S-R activo a nivel ALTO.

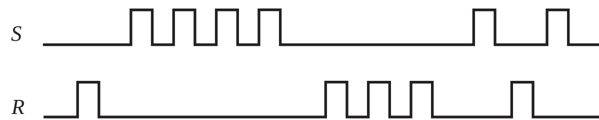


FIGURA 7.69

3. Resolver el Problema 1 para las formas de onda de entrada de la Figura 7.70.

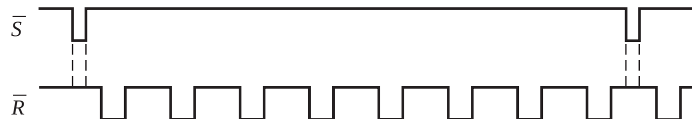


FIGURA 7.70

4. Determinar las salidas Q y $\overline{Q}$ de un *latch* S-R con entrada de habilitación para las entradas de la Figura 7.71. Dibujarlas en función de la entrada de habilitación. Suponer que, inicialmente, Q está a nivel BAJO.

5. Resolver el Problema 4 para las entradas de la Figura 7.72.

6. Resolver el Problema 4 para las entradas de la Figura 7.73.

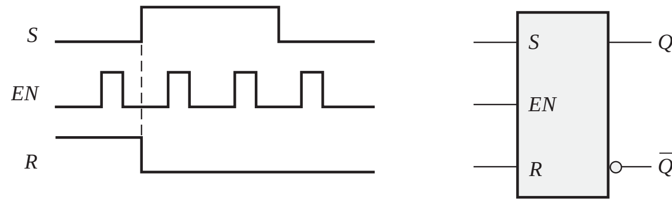


FIGURA 7.71

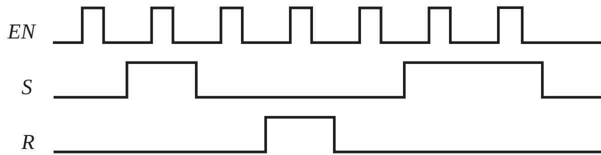


FIGURA 7.72

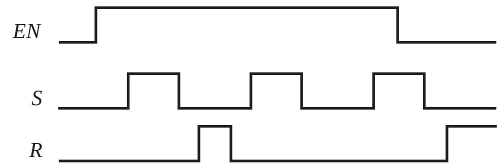


FIGURA 7.73

7. En un *latch* D con entrada de habilitación, se observan en sus entradas las formas de onda de la Figura 7.74. Dibujar el diagrama de tiempos, mostrando la forma de onda de salida que esperaríamos observar en Q si el *latch* se encuentra inicialmente en estado RESET.

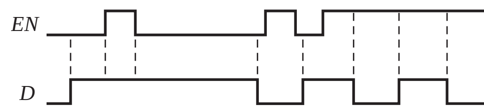


FIGURA 7.74

SECCIÓN 7.2 Flip-flops disparados por flanco

8. En la Figura 7.75 se muestran dos flip-flops S-R disparados por flanco. Si las entradas son las que se indican, dibujar la salida Q de cada flip-flop en función de la señal de reloj y explicar la diferencia entre los dos. Los flip-flops se encuentran inicialmente en estado RESET.

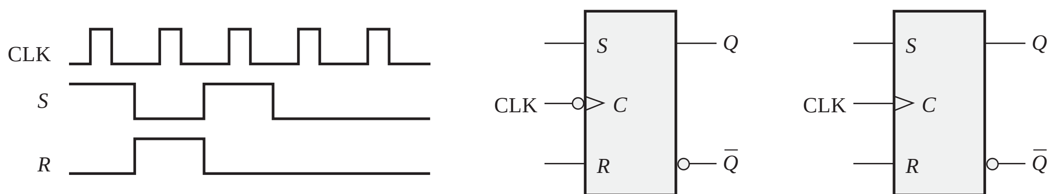


FIGURA 7.75

9. La salida Q de un flip-flop S-R disparado por flanco se muestra en la Figura 7.76 en función de la señal del reloj. Determinar las formas de onda de entrada que se necesitan en las entradas S y R para producir esta salida, si el flip-flop es de tipo disparado por flanco positivo.

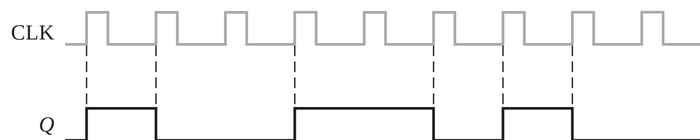


FIGURA 7.76

10. Dibujar la salida Q en función del reloj para un flip-flop D cuyas entradas son las que se muestran en la Figura 7.77. Suponer disparo por flanco positivo y que Q se encuentra inicialmente a nivel BAJO.

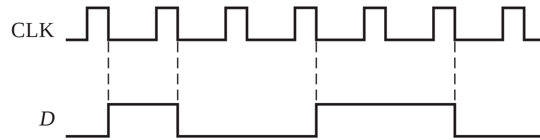


FIGURA 7.77

11. Resolver el Problema 10 para las entradas de la Figura 7.78.

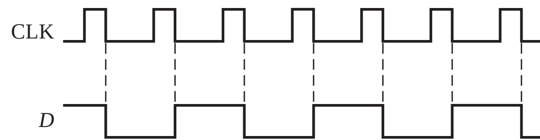


FIGURA 7.78

12. Para un flip-flop J-K disparado por flanco positivo cuyas entradas son las que se muestran en la Figura 7.79, determinar la salida Q en función del reloj. Suponer que, inicialmente, Q está a nivel BAJO.

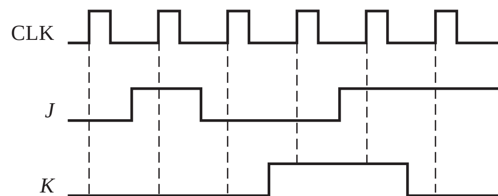


FIGURA 7.79

13. Resolver el Problema 12 para las entradas de la Figura 7.80.

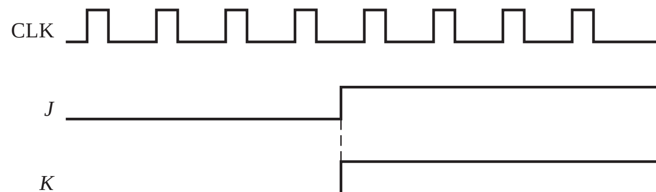


FIGURA 7.80

14. Determinar la salida Q en función del reloj si las señales que se muestran en la Figura 7.81 se aplican a las entradas de un flip-flop J-K. Suponer que Q se encuentra inicialmente a nivel BAJO.

15. Para un flip-flop J-K disparado por flanco negativo cuyas entradas son las de la Figura 7.82, desarrollar la forma de onda de salida Q en función del reloj. Suponer que Q se encuentra inicialmente a nivel BAJO.

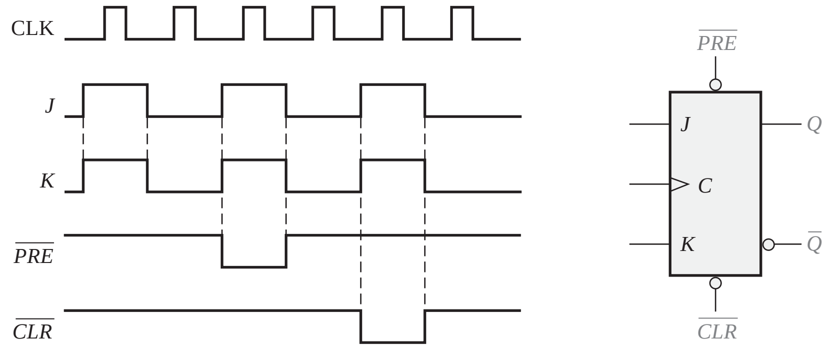


FIGURA 7.81

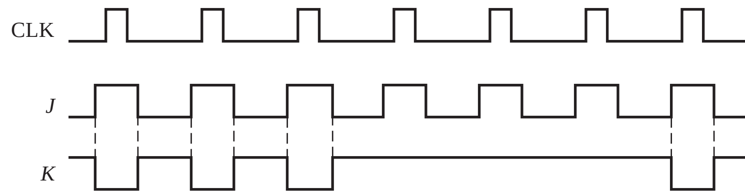


FIGURA 7.82

16. Se aplican los siguientes datos serie a un flip-flop a través de puertas AND, como se indica en la Figura 7.83. Determinar los datos serie resultantes que aparecen en la salida Q. Hay un impulso de reloj por cada periodo de bit. Suponer que, inicialmente, Q es 0 y, $\overline{PRE}$ y $\overline{CLR}$ están a nivel ALTO. Los bits de más a la derecha son los primeros que se aplican.

J_1: 1 0 1 0 0 1 1

J_2: 0 1 1 1 0 1 0

J_3: 1 1 1 1 0 0 0

K_1: 0 0 0 1 1 1 0

K_2: 1 1 0 1 1 0 0

K_3: 1 0 1 0 1 0 1

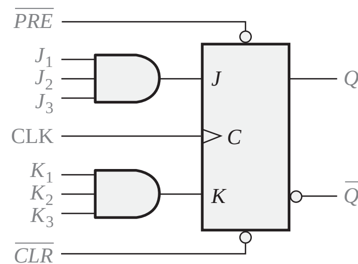


FIGURA 7.83

17. Completar el diagrama de tiempos de la Figura 7.84 para el circuito de la Figura 7.83, dibujando la salida Q que, inicialmente, está a nivel BAJO. Suponer que $\overline{PRE}$ y $\overline{CLR}$ permanecen a nivel ALTO.

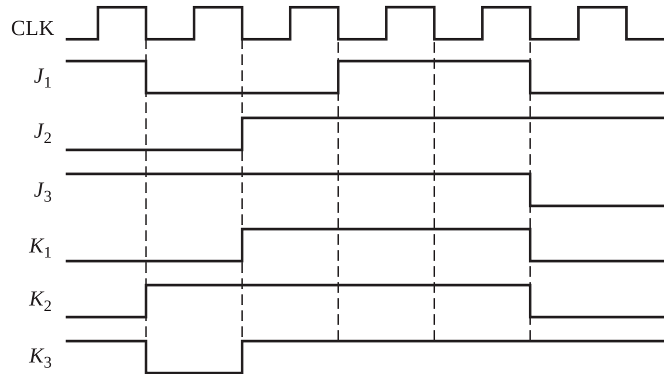


FIGURA 7.84

18. Resolver el Problema 17 con las mismas entradas *J* y *K*, pero con las entradas $\overline{PRE}$ y $\overline{CLR}$ que se muestran en la Figura 7.85 en función del reloj.

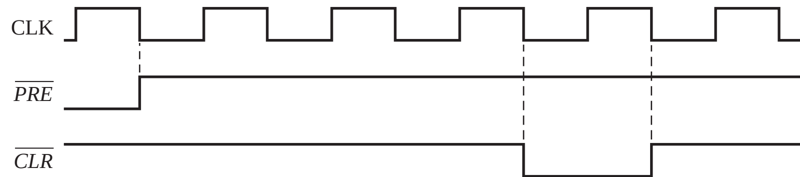


FIGURA 7.85

SECCIÓN 7.3 Características de operación de los flip-flops

19. ¿Qué determina la disipación de potencia de un flip-flop?

20. Típicamente, la hoja de características de un fabricante especifica cuatro retardos de propagación diferentes asociados con un flip-flop. Nombrar y describir cada uno de ellos.

21. La hoja de especificaciones de un determinado flip-flop especifica que la duración mínima de un nivel ALTO para cada impulso de reloj es 30 ns y que la duración mínima para un nivel BAJO es de 37 ns. ¿Cuál es la frecuencia máxima de funcionamiento?.

22. El flip-flop de la Figura 7.86 se encuentra inicialmente en RESET. Mostrar la relación entre la salida *Q* y el impulso de reloj, si el retardo de propagación t_{PLH} (del reloj a *Q*) es de 8 ns.

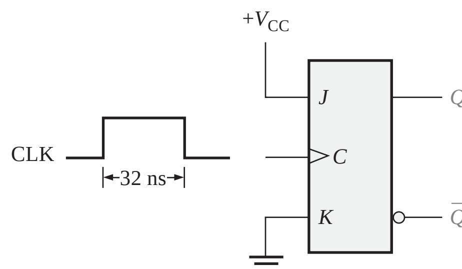


FIGURA 7.86

23. La corriente directa requerida por un determinado flip-flop que funciona a +5 V de tensión resulta ser de 10 mA. Un determinado dispositivo digital utiliza 15 de estos flip-flops.

Determinar la capacidad de corriente requerida para la fuente de continua de +5 V y la potencia total disipada por el sistema.

24. Para el circuito de la Figura 7.87, determinar la frecuencia máxima de la señal del reloj para un funcionamiento fiable, si el tiempo de *setup* (establecimiento) de cada flip-flop es de 2 ns y los retardos de propagación (t_{PLH} y t_{PHL}) del reloj a la salida son de 5 ns para cada flip-flop.

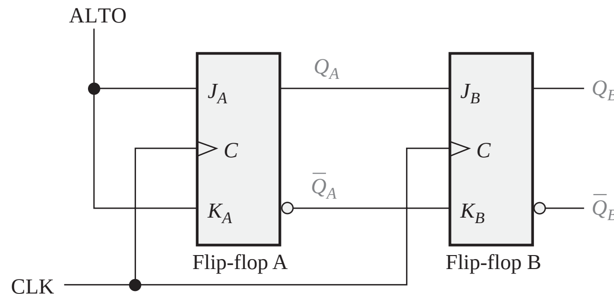


FIGURA 7.87

SECCIÓN 7.4 Aplicaciones básicas de los flip-flops

25. Un flip-flop D se encuentra conectado como se muestra en la Figura 7.88. Determinar la salida Q en función del reloj. ¿Cuál es la función que realiza este dispositivo?

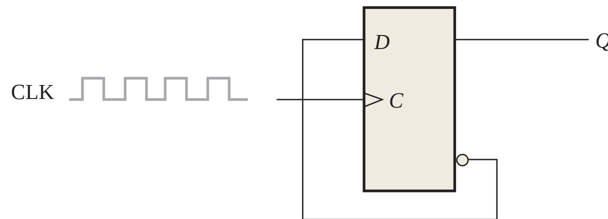


FIGURA 7.88

26. Para el circuito de la Figura 7.87, desarrollar un diagrama de tiempos para ocho impulsos de reloj, mostrando las salidas Q_A y Q_B en función del reloj.

SECCIÓN 7.5 Monoestables

27. Determinar la anchura del impulso de un monoestable 74121, si la resistencia externa es de 3,3 kΩ y el condensador externo vale 2000 pF.

28. Se quiere generar un impulso de salida de 5 μs de duración con un monoestable 74LS122. Utilizando un condensador de 10.000 pF, determinar el valor de la resistencia externa requerida.

SECCIÓN 7.6 El temporizador 555

29. Diseñar un monoestable utilizando un temporizador 555 para producir un impulso de salida de 0,25 segundos.

30. Se configura un temporizador 555 para funcionar como multivibrador aestable, como se muestra en la Figura 7.89. Determinar su frecuencia.

31. Determinar los valores de las resistencias externas de un temporizador 555 utilizado como multivibrador aestable con frecuencia de salida de 20 kHz, si el condensador C vale 0,002 μF y el ciclo de trabajo es del 75 % aproximadamente.

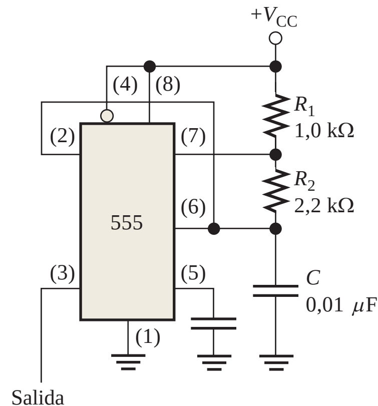


FIGURA 7.89

SECCIÓN 7.7 Localización de averías

32. Se prueba el flip-flop de la Figura 7.90 bajo todas las posibles condiciones de entrada, tal como se muestra. ¿Está funcionando correctamente? Si no es así ¿cuál es la causa de fallo más probable?

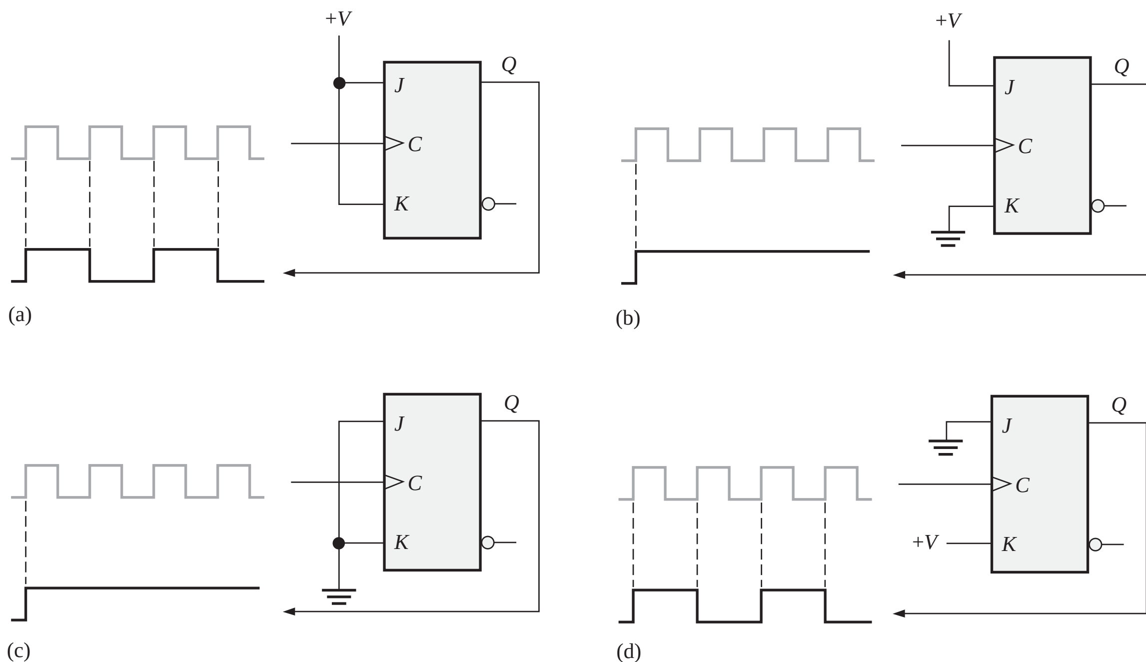


FIGURA 7.90

33. Se utiliza una cuádruple puerta NAND 74HC00 para construir un *latch* S-R con entrada de habilitación en un prototipo de tarjeta de laboratorio, como muestra la Figura 7.91. El esquema de la parte (a) se utiliza para conectar el circuito de la parte (b). Cuando intentamos poner en funcionamiento el *latch*, nos encontramos con que la salida Q permanece a nivel ALTO independientemente de los valores de las entradas. Determinar cuál es el problema.

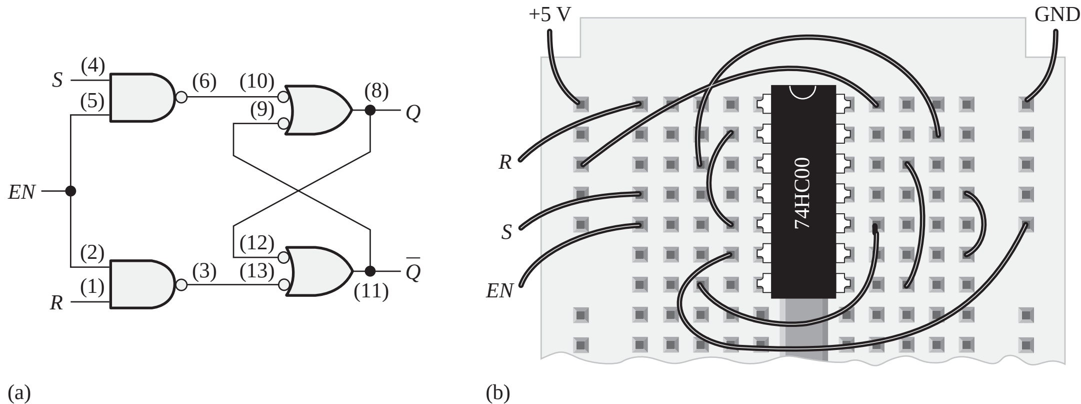


FIGURA 7.91

34. Determinar si el flip-flop de la Figura 7.92 está funcionando adecuadamente y, en caso contrario, identificar el fallo más probable.

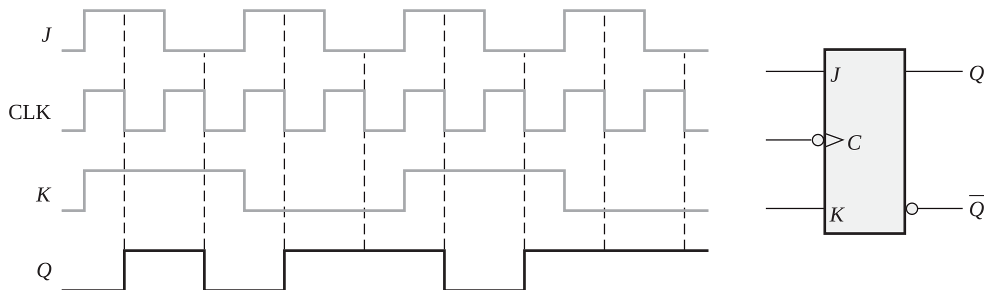


FIGURA 7.92

35. El circuito paralelo de almacenamiento de datos de la Figura 7.36 no funciona adecuadamente. Para depurarlo, primero nos aseguramos de que V_{CC} y tierra se encuentran conectados y, luego, aplicamos niveles BAJOS a todas las entradas *D* e introducimos impulsos en la línea del reloj. Se comprueba que las salidas *Q* están todas a nivel BAJO, por lo que por el momento todo es correcto. A continuación se aplican niveles ALTOS a todas las entradas *D* y de nuevo se introducen impulsos en la línea del reloj. Cuando comprobamos las salidas *Q*, todavía permanecen a nivel BAJO. ¿Cuál es el problema y cuál sería el procedimiento que utilizaríamos para aislar el fallo a un único dispositivo?

36. El circuito del flip-flop de la Figura 7.93(a) se utiliza para generar una secuencia de cuenta binaria. Las puertas forman un decodificador que se supone que produce un nivel ALTO cuando ocurre un cero binario o el estado tres binario (00 u 11). Analizando las salidas Q_A y Q_B, se obtiene la imagen mostrada en la parte (b), que revela *glitches* en la salida del decodificador (*X*), además de los impulsos correctos. ¿Qué es lo que causa estos *glitches* y cómo se pueden eliminar?

37. Determinar las salidas Q_A, Q_B y *X* durante seis impulsos de reloj en la Figura 7.93(a) para cada uno de los siguientes fallos en circuitos TTL. Inicialmente Q_A y Q_B están a nivel BAJO.

(a) La entrada J_A está en circuito abierto.

(b) La entrada K_B está en circuito abierto.



(a)



(b)

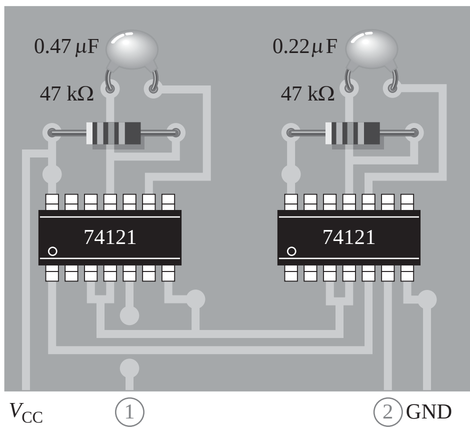
FIGURA 7.93

(c) La entrada Q_B está en circuito abierto.

(d) La entrada de reloj en el flip-flop B está cortocircuitada.

(e) La puerta G_2 está en circuito abierto.

38. Se conectan dos monoestables 74121 en una tarjeta, como se muestra en la Figura 7.94. Tras observar la pantalla del osciloscopio, ¿sacaríamos la conclusión de que el circuito está funcionando adecuadamente? En caso contrario, ¿cuál es la causa más probable del fallo?



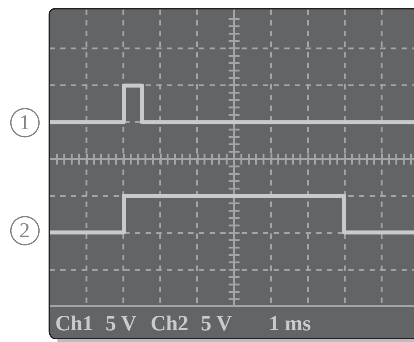


FIGURA 7.94



Aplicación a los sistemas digitales

39. Utilizar temporizadores 555 para implementar los monoestables de 4 y 25 segundos para los circuitos de temporización del sistema de control de los semáforos. La entrada de disparo del 555 no puede permanecer a nivel BAJO después de una transición negativa, de forma que tenemos que desarrollar un circuito para producir impulsos negativos muy cortos, con el fin de disparar los temporizadores corto y largo cuando el sistema pasa por cada estado.



Problemas especiales de diseño

40. Diseñar un circuito contador básico que genere una secuencia binaria de cero a siete, utilizando flip-flops J-K disparados por flanco negativo.

41. En el departamento de logística de una fábrica de pelotas, éstas ruedan por una cinta y por una rampa hasta llegar a una caja. Cada pelota que pasa por la rampa activa un conmutador que produce un impulso eléctrico. La capacidad de cada caja es de 32 pelotas. Diseñar un circuito lógico para indicar cuándo una caja está llena, de forma que pueda ser sustituida por otra vacía.

42. Enumerar los cambios que serían necesarios en el sistema de control de semáforos para añadir una indicación de giro a la derecha de 15 segundos de duración en la calle principal. La indicación aparecería después de la luz roja y antes de la verde. Modificar el diagrama de estados del Capítulo 6 de manera que refleje estos cambios.

RESPUESTAS

REVISIONES DE CADA SECCIÓN

SECCIÓN 7.1 ***Latches***

1. Tres tipos de *latches* son el S-R, el S-R con entrada de habilitación y el D con entrada de habilitación.
2. $SR = 00$, NC; $SR = 01$, $Q = 0$; $SR = 10$, $Q = 1$; $SR = 11$, no válido
3. $Q = 1$

SECCIÓN 7.2 **Flip-flops disparados por flanco**

1. La salida de un *latch* S-R con entrada de habilitación puede cambiar siempre que la entrada de habilitación (*EN*) esté activa. La salida de un flip-flop S-R disparado por flanco puede cambiar sólo durante los flancos de disparo de un impulso de reloj.
2. El flip-flop J-K no tiene ningún estado no válido, como ocurre con el flip-flop S-R.
3. La salida Q se pone a nivel ALTO durante el flanco posterior del primer impulso del reloj, se pone a nivel BAJO durante el flanco posterior del segundo impulso, a nivel ALTO en el flanco posterior del tercer impulso y a nivel BAJO en el flanco posterior del cuarto impulso.

SECCIÓN 7.3 **Características de funcionamiento de los flip-flops**

1. **(a)** El tiempo de *setup* (establecimiento) es el tiempo que los datos de entrada deben estar presentes antes del flanco de disparo del impulso de reloj.

   (b) Tiempo de *hold* (mantenimiento) es el tiempo que los datos deben permanecer en la entrada después del flanco de disparo del impulso de reloj.
2. El 74AHC74 puede funcionar a la frecuencia máxima, de acuerdo con la Tabla 7.5.

SECCIÓN 7.4 **Aplicaciones de los flip-flops**

1. Un registro es un grupo de flip-flops de almacenamiento de datos.

2. Para funcionar como divisor por dos, el flip-flop tiene que estar en modo de basculación (J = 1, K =1).
3. Se necesitan seis flip-flops para formar un divisor por 64.

SECCIÓN 7.5 Monoestables

1. Un monoestable no redisparable no puede responder a otra entrada de disparo mientras que se encuentra en su estado inestable. Un monoestable disparable responde a cada entrada de disparo.
2. La anchura de los impulsos se ajusta mediante componentes R y C externos.

SECCIÓN 7.6 El temporizador 555

1. Un aestable no tiene estados estables. Un monoestable tiene un estado estable.
2. Ciclo de trabajo = (15 ms/20 ms) 100% = 75%

SECCIÓN 7.7 Localización de averías

1. Sí, se puede utilizar un flip-flop D disparado por flanco negativo.
2. Se puede utilizar un temporizador 555 funcionando en modo de multivibrador aestable como reloj.

PROBLEMAS RELACIONADOS

7.1 La salida Q es la misma que la mostrada en la Figura 7.5(b).

7.2 *Véase* la Figura 7.95.

7.3 *Véase* la Figura 7.96.

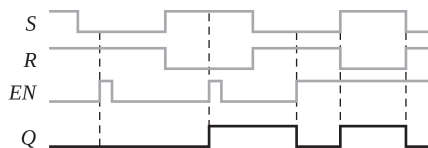


FIGURA 7.95

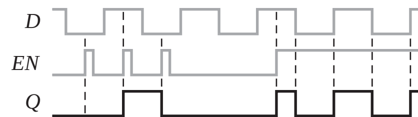


FIGURA 7.96

7.4 *Véase* la Figura 7.97.

7.5 *Véase* la Figura 7.98.

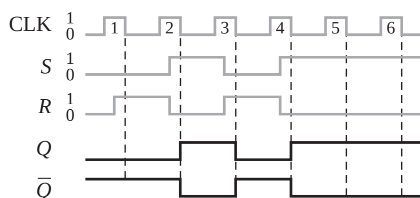


FIGURA 7.97

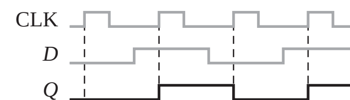


FIGURA 7.98

7.6 *Véase* la Figura 7.99.

7.7 *Véase* la Figura 7.100.

7.8 *Véase* la Figura 7.101.

7.9 *Véase* la Figura 7.102.

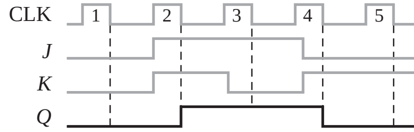


FIGURA 7.99

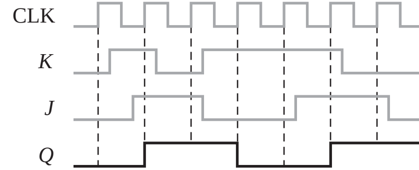


FIGURA 7.100

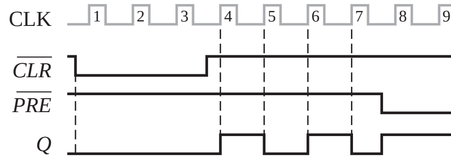


FIGURA 7.101

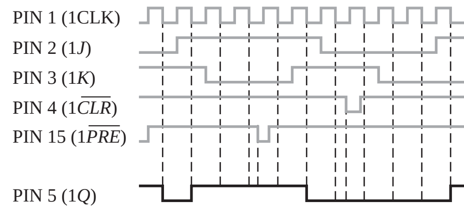


FIGURA 7.102

7.10 $2^5 = 32$. Se requieren 5 flip-flops.

7.11 Dieciséis estados requieren cuatro flip-flops ($2^4 = 16$).

7.12 $C_{EXT} = 7143$ pF conectado desde *CX* a *RX/CX* del 74142.

7.13 $C_{EXT} = 560$ pF, $R_{EXT} = 27$ kΩ. *Véase* la Figura 7.103.

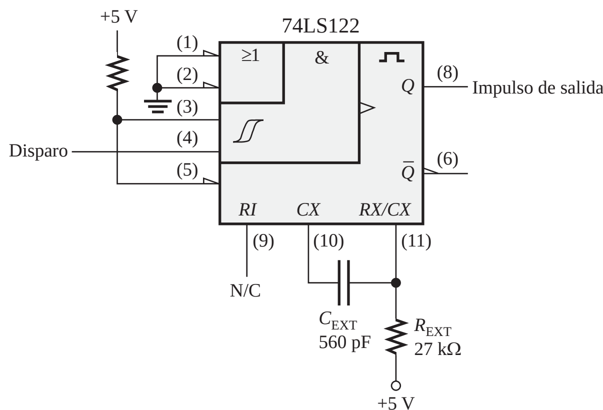


FIGURA 7.103

7.14 $R_1 = 91$ kΩ

7.15 Ciclo de trabajo ≅ 32%

AUTOTEST

1. (a) **2.** (c) **3.** (d) **4.** (b) **5.** (d) **6.** (d)
7. (a) **8.** (b) **9.** (d) **10.** (d) **11.** (c) **12.** (f)

8

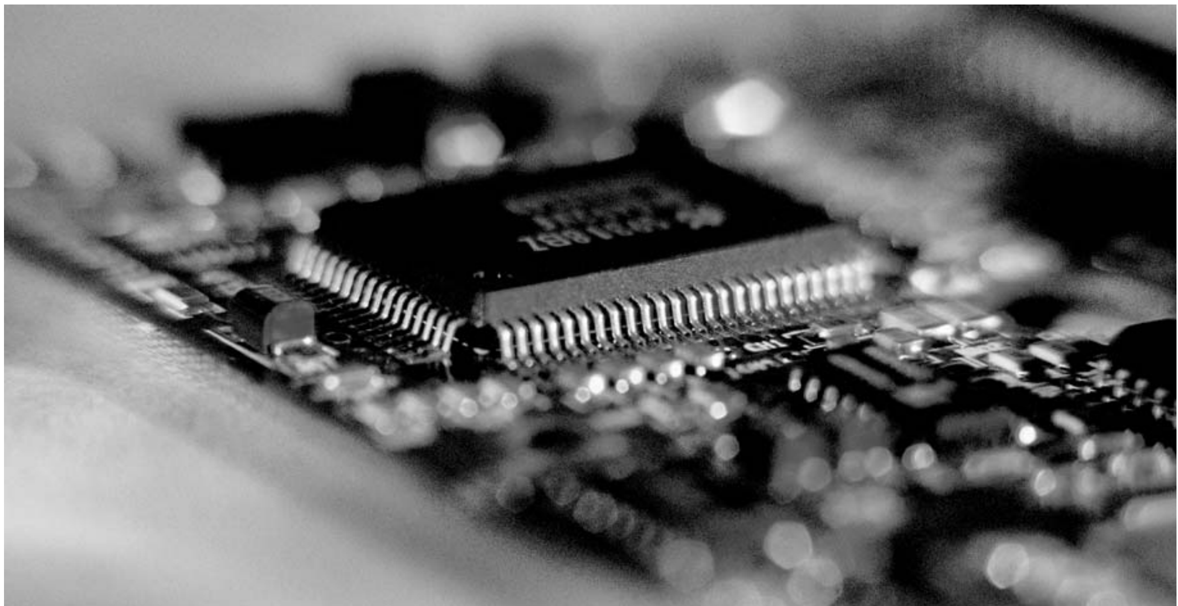
CONTADORES

CONTENIDO DEL CAPÍTULO

8.1 Funcionamiento del contador asíncrono
8.2 Funcionamiento del contador síncrono
8.3 Contador síncrono ascendente/descendente
8.4 Diseño de contadores síncronos
8.5 Contadores en cascada
8.6 Decodificación de contadores
8.7 Aplicaciones de los contadores
8.8 Símbolos lógicos con notación de dependencia
8.9 Localización de averías
■ ■ ■ Aplicación a los sistemas digitales

OBJETIVOS DEL CAPÍTULO

- Describir la diferencia entre un contador asíncrono y un contador síncrono.
- Analizar los diagramas de tiempos de los contadores.
- Analizar los circuitos contadores.
- Explicar en qué afecta el retardo de propagación al funcionamiento de un contador.
- Determinar el módulo de un contador.
- Modificar el módulo de un contador.
- Establecer las diferencias entre contadores binarios de 4 bits y contadores de décadas.



- Utilizar un contador ascendente/descendente para generar secuencias binarias directas e inversas.
- Determinar la secuencia de un contador.
- Utilizar circuitos integrados contadores en diversas aplicaciones.
- Diseñar un contador que pueda tener cualquier secuencia de estados especificada.
- Conectar varios contadores en cascada para conseguir módulos mayores.
- Utilizar puertas lógicas para decodificar cualquier estado de un contador.
- Eliminar los *glitches* en la decodificación de contadores.
- Explicar cómo funciona un reloj digital.
- Interpretar los símbolos lógicos de los contadores que usan notación de dependencia.
- Localizar averías en los contadores y resolver los distintos tipos de fallos.

PALABRAS CLAVE

- Asíncrono
- Nuevo ciclo (*recycle*)
- Módulo
- Década
- Síncrono
- Fin de cuenta
- Máquina de estados
- Diagrama de estados
- Conexión en cascada

INTRODUCCIÓN

Como ya se ha visto en el Capítulo 8, los flip-flops pueden conectarse entre sí para realizar funciones de recuento. A esta combinación de flip-flops se la denomina contador. El número de flip-flops que se utilizan y la forma en que se conectan determinan el número de estados (que recibe el nombre de módulo) y también la secuencia específica de estados por los que pasa el contador durante un ciclo completo.

Dependiendo del modo en que se aplique la señal de reloj, los contadores se clasifican en dos amplias categorías: asíncronos y síncronos. En los contadores asíncronos, normalmente denominados *contadores con propagación (ripple counters),* se aplica una señal de reloj externa a la entrada de reloj del primer flip-flop y luego a los siguientes flip-flops se les aplica la señal de reloj mediante la salida del flip-flop anterior. En los contadores síncronos, la entrada de reloj se conecta a todos los flip-flops, de forma que se les aplica la señal de reloj simultáneamente. Dentro de cada una de estas dos categorías, los contadores se clasifican por el tipo de secuencia, el número de estados o el número de flip-flops del contador.

DISPOSITIVOS DE FUNCIÓN FIJA

74XX93	74XX161	74XX162
74XX163	74XX190	74XX47

■■■ APLICACIÓN A LOS SISTEMAS DIGITALES

Esta aplicación a los sistemas digitales ilustra los conceptos que se tratan en el capítulo. Se continúa con el sistema de control de semáforos de los últimos dos capítulos. Este capítulo se ocupa de la lógica secuencial del sistema que produce la secuencia de luces basada en las entradas de los circuitos de temporización y del sensor de vehículos. Las partes desarrolladas en los Capítulo 6 y 7 se combinan con la lógica secuencial para completar el sistema.

8.1 FUNCIONAMIENTO DEL CONTADOR ASÍNCRONO

El término ***asíncrono*** se refiere a los sucesos que no poseen una relación temporal fija entre ellos y que, generalmente, no ocurren al mismo tiempo. Un **contador asíncrono** es aquél en el que los flip-flops (FF) del contador no cambian de estado exactamente al mismo tiempo, dado que no comparten el mismo impulso de reloj.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Describir el funcionamiento de un contador asíncrono binario de 2 bits. ■ Describir el funcionamiento de un contador asíncrono binario de 3 bits. ■ Definir la *propagación* en contadores asíncronos. ■ Describir el funcionamiento de un contador de décadas asíncrono. ■ Desarrollar los diagramas de tiempos de los contadores. ■ Describir el contador asíncrono binario de 4 bits 74LS93.

Contador asíncrono binario de 2 bits

▲ *La entrada de reloj de un contador asíncrono siempre está conectada sólo al flip-flop LSB.*

La Figura 8.1 presenta un contador de 2 bits conectado para que funcione en modo asíncrono. Observe que el reloj (CLK) está conectado únicamente a la entrada de reloj (*C*) del primer flip-flop, FF0. El segundo flip-flop, FF1, se dispara mediante la salida $\overline{Q}_0$ de FF0. FF0 cambia de estado durante el flanco positivo de cada impulso de reloj, pero FF1 sólo cambia cuando es disparado por una transición positiva de la salida $\overline{Q}_0$ de FF0. Debido al retardo de propagación inherente al paso de las señales por un flip-flop, las transiciones de los impulsos de entrada del reloj y de la salida $\overline{Q}_0$ de FF0 no pueden ocurrir nunca al mismo tiempo. Por tanto, los dos flip-flops nunca se disparan de forma simultánea, por lo que el modo de funcionamiento de este contador es asíncrono.

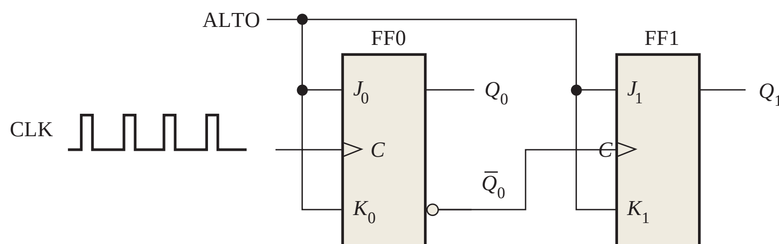


FIGURA 8.1 Contador asíncrono binario de 2 bits.

Diagrama de tiempos. Vamos a examinar el funcionamiento básico del contador asíncrono de la Figura 8.1, aplicando cuatro impulsos de reloj a FF0 y observando la salida *Q* de cada flip-flop. La Figura 8.2 ilustra los cambios de estado en las salidas del flip-flop en respuesta a los impulsos de reloj. Ambos flip-flops están conectados en modo de basculación ($J = 1$, $K = 1$) y se presupone que, inicialmente, están en estado RESET (*Q* a nivel BAJO).

El flanco positivo de CLK1 (impulso de reloj 1) hace que la salida Q_0 de FF0 pase a nivel ALTO, como se muestra la Figura 8.2. Al mismo tiempo, la salida $\overline{Q}_0$ pasa a nivel BAJO, pero esto no afecta a FF1, ya que tiene que ser una transición positiva la que le dispare. Después del flanco anterior de CLK1, $Q_0 = 1$ y $Q_1 = 0$.

▲ *Los contadores asíncronos se conocen también como contadores con propagación.*

El flanco positivo de CLK2 hace que Q_0 pase a nivel BAJO. La salida $\overline{Q}_0$ se pone a nivel ALTO y dispara FF1, haciendo que Q_1 pase a nivel ALTO. Tras el flanco anterior de CLK2, $Q_0 = 0$ y $Q_1 = 1$. El flanco positivo de CLK3 hace que Q_0 pase a nivel ALTO de nuevo. La salida $\overline{Q}_0$ se pone a nivel BAJO y no afecta al estado de FF1. Por tanto, tras el flanco anterior de CLK3, $Q_0 = 1$ y $Q_1 = 1$. El flanco positivo de

CLK4 hace que Q_0 pase a nivel BAJO, mientras que $\overline{Q}_0$ se pone a nivel ALTO y dispara FF1, haciendo que Q_1 pase a nivel BAJO. Después del flanco anterior de CLK4, $Q_0 = 0$ y $Q_1 = 0$. El contador ha vuelto a su estado original (los dos flip-flops se encuentran en estado RESET).

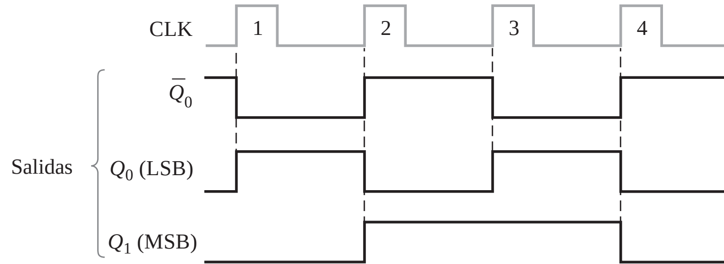


FIGURA 8.2 Diagrama de tiempos del contador de la Figura 8.1. Como en los capítulos anteriores, las formas de onda de salida se muestran en negro.

En el diagrama de tiempos, las formas de onda de las salidas Q_0 y Q_1 se muestran en función de los impulsos de reloj, como ilustra la Figura 8.2. Para simplificar, las transiciones de Q_0, Q_1 y los impulsos de reloj se muestran como simultáneos, aunque se trate de un contador asíncrono. Existe, por supuesto, un ligero retardo entre las transiciones de CLK y Q_0 y las transiciones de $\overline{Q}_0$ y Q_1.

▲ *En la lógica digital, Q_0 es siempre el bit menos significativo, (LSB), a menos que se indique lo contrario.*

Observe en la Figura 8.2, que el contador de 2 bits dispone de cuatro estados diferentes, como cabría esperar de dos flip-flops ($2^2 = 4$). Además, téngase en cuenta que si Q_0 representa el bit menos significativo (LSB) y Q_1 representa el bit más significativo (MSB), la secuencia de los estados del contador representa una secuencia de números binarios, como se muestra en la Tabla 8.1.

Puesto que pasa por una **secuencia** binaria, el contador de la Figura 8.1 es un contador binario. En realidad, cuenta el número de impulsos de reloj hasta el tercero y, en el cuarto impulso, inicia un nuevo ciclo a partir de su estado original ($Q_0 = 0$, $Q_1 = 0$). El inicio de un **nuevo ciclo** (*recycle*, término que se aplica comúnmente al funcionamiento de los contadores) se refiere a la transición del contador de su estado final a su estado original.

Contador asíncrono binario de 3 bits La secuencia de estados de un contador binario de 3 bits se presenta en la Tabla 8.2 y en la Figura 8.3(a) se muestra un contador asíncrono binario de 3 bits. Su funcionamiento básico es el mismo que el del contador de 2 bits, excepto en que el contador de 3 bits tiene ocho estados, ya que está formado por tres flip-flops. En la Figura 8.3(b) se presenta un diagrama de tiempos para ocho impulsos de reloj. Observe que el contador de la Figura 8.3 avanza a través de una secuencia binaria desde cero hasta siete, iniciando después un nuevo ciclo desde su estado cero. Este contador puede ampliarse fácilmente a un contador mayor, conectando flip-flops adicionales.

Impulso de reloj	Q_1	Q_2
Inicialmente	0	0
1	0	1
2	1	0
3	1	1
4 (nuevo ciclo)	0	0

TABLA 8.1 Secuencia de estados binarios para el contador de la Figura 8.1.

Impulso de reloj	Q_2	Q_1	Q_0
Inicialmente	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1
8 (nuevo ciclo)	0	0	0

TABLA 8.2 Secuencia de estados de un contador binario de tres bits.

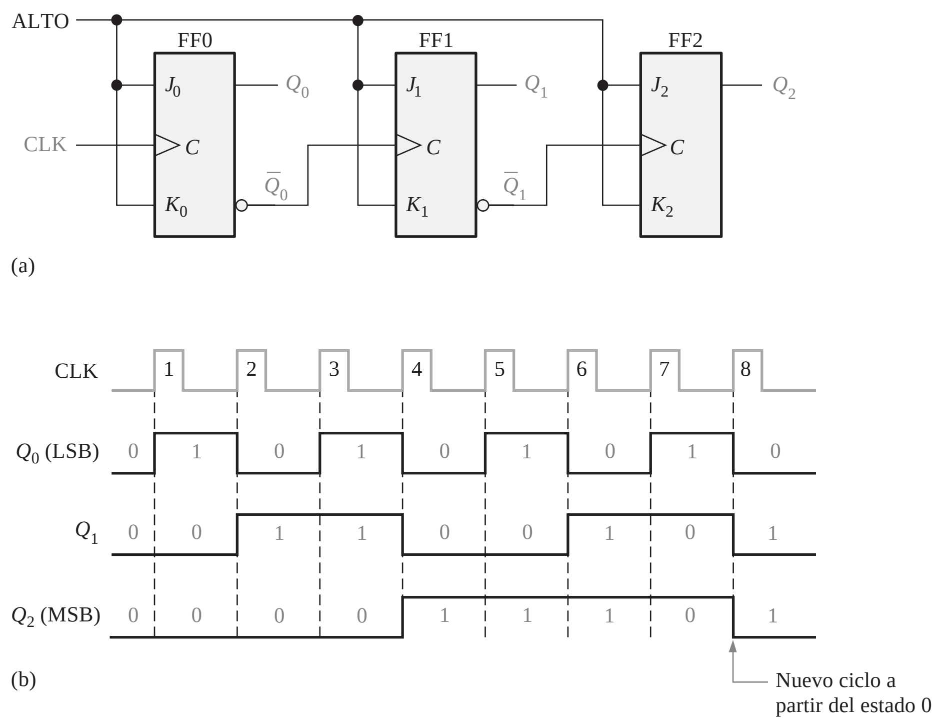


FIGURA 8.3 Contador binario asíncrono de tres bits y su diagrama de tiempos para un ciclo.

Retardo de propagación A los contadores asíncronos también se les denomina **contadores con propagación** por la siguiente razón: el efecto de un impulso en la entrada de reloj lo "siente" primero FF0. Este efecto no llega inmediatamente a FF1, debido al retardo de propagación a través de FF0. Del mismo modo, se produce un retardo de propagación a través de FF1, antes de que FF2 pueda ser disparado. Por tanto, el efecto de un impulso en la entrada de reloj se "propaga" a través del contador, tardando un cierto tiempo en alcanzar el último de los flip-flops, debido a los retardos de propagación.

Como ilustración, observe que todos los flip-flops del contador de la Figura 8.3 cambian de estado a en el flanco anterior de CLK4. Este efecto de propagación de la señal de reloj se muestra en la Figura 8.4 para los

cuatro primeros impulsos de reloj, indicando los retardos de propagación. La transición de nivel ALTO a nivel BAJO de Q_0 se produce después de un determinado retardo (t_{PHL}) después de la transición positiva del impulso de reloj. La transición de nivel ALTO a nivel BAJO de Q_1 ocurre un tiempo (t_{PLH}) después de la transición positiva de $\overline{Q}_0$. La transición de nivel BAJO a nivel ALTO de Q_2 se produce después de otra unidad de retardo (t_{PLH}) después de la transición positiva de $\overline{Q}_1$. Como puede ver, FF2 no se dispara hasta que han transcurrido dos unidades de retardo después del flanco positivo del impulso de reloj, CLK4. Por tanto, se necesitan tres unidades de retardo para que el efecto del impulso de reloj CLK4 se propague a través del contador y Q_2 pase de nivel BAJO a nivel ALTO.

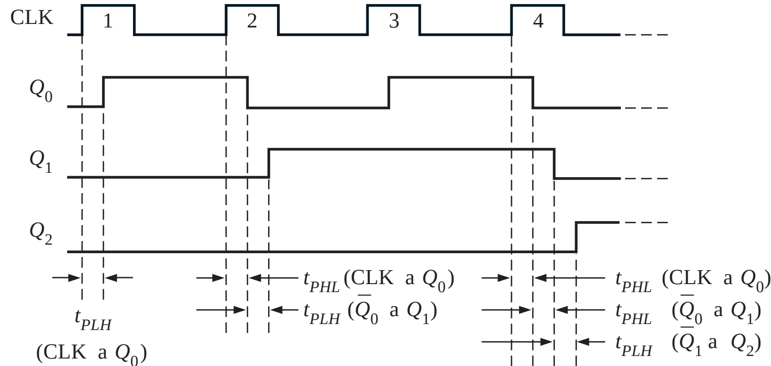


FIGURA 8.4 Retardos de propagación en un contador binario asíncrono (con propagación de reloj) de 3 bits.

Este retardo acumulativo de un contador asíncrono es una de sus mayores desventajas para muchas aplicaciones, ya que limita la velocidad a la que el contador puede ser sincronizado, y puede dar lugar a problemas de decodificación. El retardo acumulativo máximo en un contador tiene que ser menor que el período de la señal de reloj.

EJEMPLO 8.1

En la Figura 8.5(a) se muestra un contador asíncrono binario de 4 bits. Cada flip-flop es disparado por flanco negativo y tiene un retardo de propagación de 10 nanosegundos (ns). Dibujar un diagrama de tiempos que muestre la salida Q de cada uno de los flip-flops y determinar el retardo de propagación total desde el flanco de disparo de un impulso de reloj hasta que pueda producirse el cambio correspondiente en el estado de Q_3. Determinar también la frecuencia máxima de reloj a la que puede funcionar el contador.

Solución En la Figura 8.5(b) se muestra el diagrama de tiempos, habiendo omitido los retardos. Por lo que se refiere al retardo total, el efecto de CLK8 o CLK16 se tiene que propagar a través de cuatro flip-flops antes de que Q_3 cambie, de forma que:

$$t_{p(tot)} = 4 \times 10\,\text{ns} = \mathbf{40\,ns}$$

La frecuencia máxima de reloj es:

$$f_{\max} = \frac{1}{t_{p(tot)}} = \frac{1}{40\,\text{ns}} = \mathbf{25\,MHz}$$

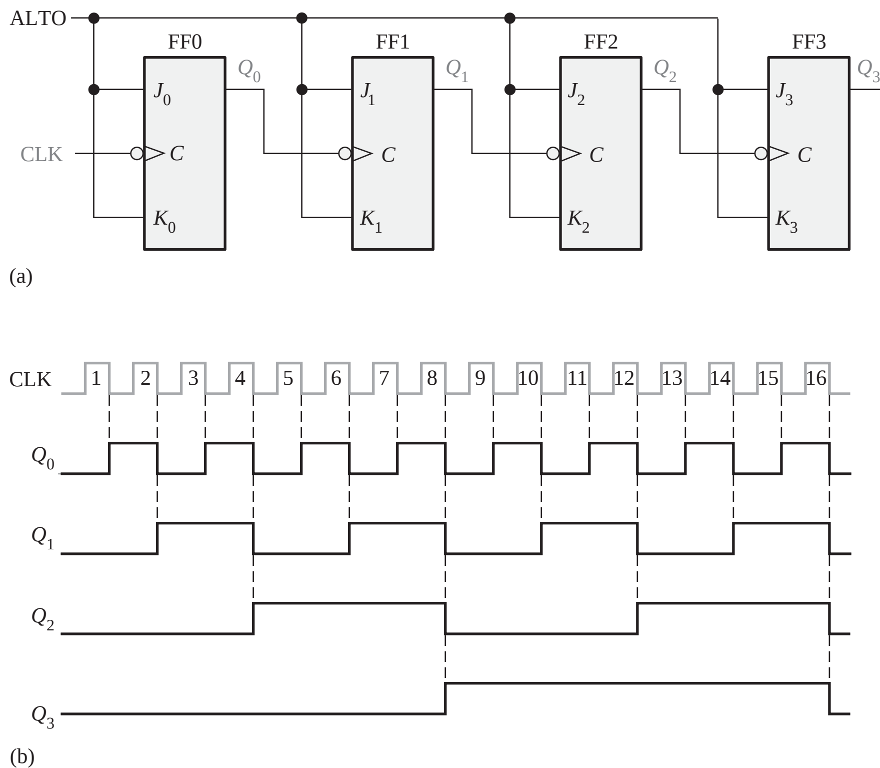


FIGURA 8.5 Contador binario asíncrono de 4 bits y su diagrama de tiempos.

Problema relacionado* Dibujar el diagrama de tiempos si todos los flip-flops de la Figura 8.5(a) fueran disparados por flanco positivo.

* Las respuestas se encuentran al final del capítulo.

Contador de décadas asíncrono

▲ *Un contador puede tener 2^n estados, siendo n el número de flip-flops.*

El ***módulo*** de un contador es el número de estados distintos por el que el contador puede pasar de forma secuencial. El número máximo de posibles estados (módulo máximo) de un contador es 2^n, donde n representa el número de flip-flops del contador. También se pueden diseñar contadores que tengan un número de estados en su secuencia que sea menor que el máximo de 2^n. La secuencia resultante se denomina *secuencia truncada*.

Un módulo típico en los contadores con secuencia truncada es diez (denominado MOD10). Los contadores que tienen diez estados en su secuencia se denominan contadores de **décadas**. Un contador de décadas, cuya secuencia de cuenta vaya de cero (0000) a nueve (1001), es un contador de décadas BCD, ya que su secuencia de diez estados corresponde al código BCD. Este tipo de contadores resulta muy útil en las aplicaciones de displays, en las que se necesitan códigos BCD para la conversión a código decimal.

Para obtener una secuencia truncada, es necesario forzar al contador a que inicie un nuevo ciclo antes de haber pasado por todos los estados normales. Por ejemplo, el contador BCD de décadas tiene que comenzar

de nuevo en el estado 0000 después de pasar por el estado 1001. Un contador de décadas requiere cuatro flip-flops (tres serían insuficientes, ya que $2^3 = 8$).

Para ilustrar el principio de los contadores truncados, vamos a utilizar un contador asíncrono de 4 bits, como el del Ejemplo 8.1, en el que modificaremos su secuencia. Una manera de hacer que un contador inicie un nuevo ciclo después de haber llegado a nueve (1001) consiste en decodificar el diez (1010) con una puerta NAND y conectar la salida de la puerta NAND a las entradas de borrado ($\overline{CLR}$) de los flip-flops, como se muestra en la Figura 8.6(a).

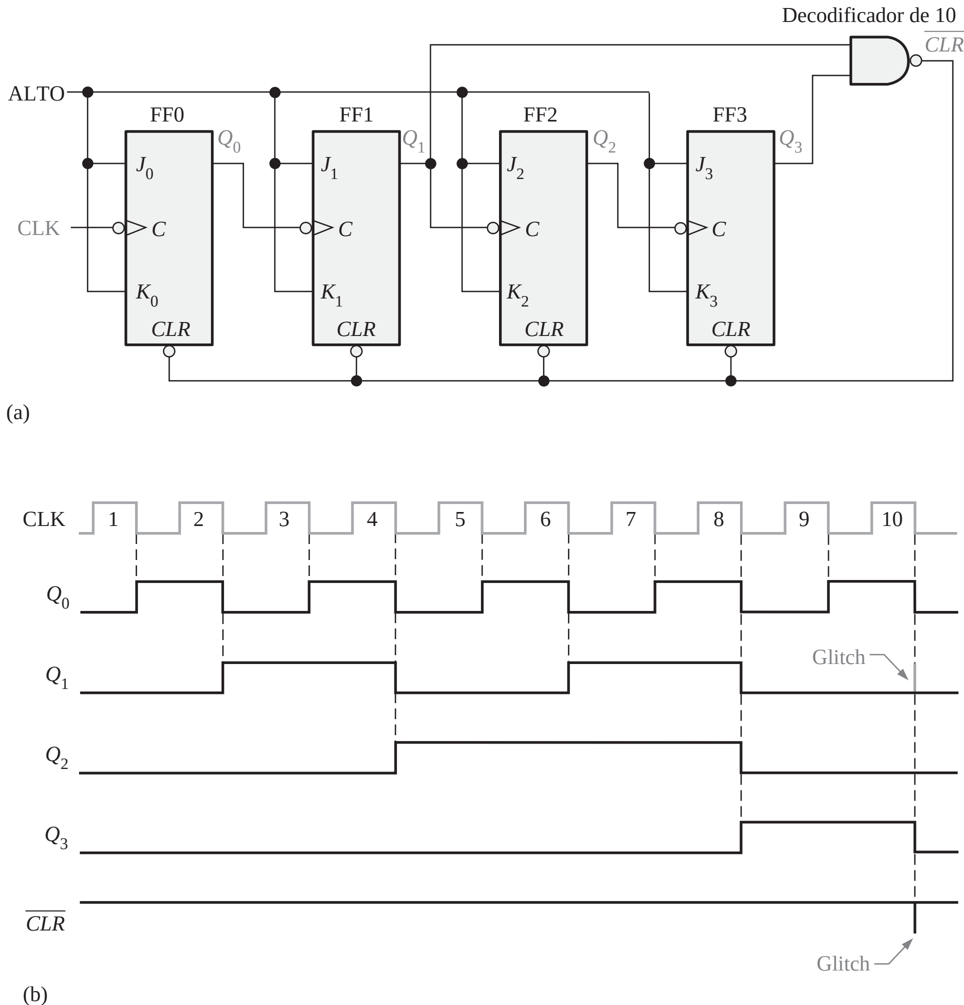


FIGURA 8.6 Contador de décadas asíncrono con reinicialización asíncrona.

Decodificación parcial Observe en la Figura 8.6(a) que únicamente Q_1 y Q_3 están conectadas a las entradas de la puerta NAND. Esta disposición es un ejemplo de *decodificación parcial*, mediante la cual dos únicos estados ($Q_1 = 1$ y $Q_3 = 1$) son suficientes para decodificar el valor diez, ya que ninguno de los otros estados (de cero a nueve) tienen Q_1 y Q_3 a nivel ALTO al mismo tiempo. Cuando el contador llega al número diez (1010), la salida de la puerta decodificadora pasa a nivel BAJO y pone a cero asíncronamente todos los flip-flops.

El diagrama de tiempos resultante se muestra en la Figura 8.6(b). Observe que hay un *glitch* en la forma de onda Q_1. La razón de este *glitch* es que Q_1 tiene que pasar primero por el nivel ALTO antes de que el número diez pueda decodificarse. Hasta unos nanosegundos después de que el contador llegue al número diez, la salida de la puerta decodificadora no se pone a nivel BAJO (las dos entradas están a nivel ALTO). Por tanto, el contador se encuentra en el estado 1010 durante un período de tiempo corto antes de que se reinicialice a 0000, produciendo, por tanto, el *glitch* en Q_1 y en la línea $(\overline{CLR})$ que sirve para poner a cero el contador.

Como muestra el Ejemplo 8.2, se pueden implementar otras secuencias truncadas de manera similar.

EJEMPLO 8.2

Explicar cómo se puede implementar un contador asíncrono que tenga módulo doce con una secuencia binaria directa desde 0000 hasta 1011.

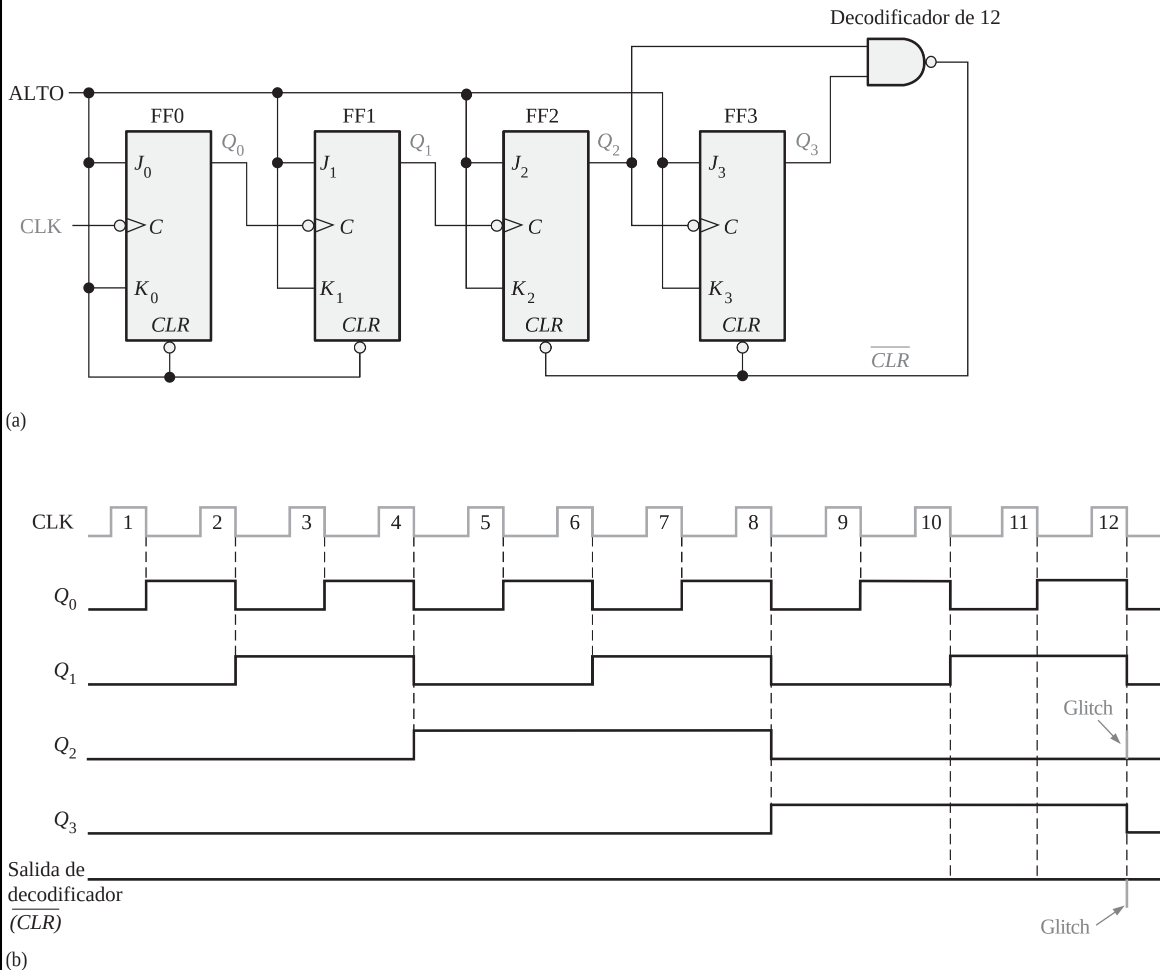


FIGURA 8.7 Contador de módulo 12 con temporización y reinicialización asíncronas.

Solución Puesto que tres flip-flops pueden generar un máximo de ocho estados, necesitamos cuatro flip-flops para producir cualquier módulo mayor que ocho y menor o igual que dieciséis.

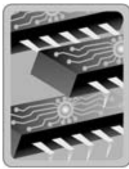
Cuando el contador alcanza el estado final 1011 tiene que iniciar un nuevo ciclo a partir de 0000, en lugar de pasar al siguiente estado natural, 1100, como ilustra la siguiente secuencia:

Q_3	Q_2	Q_1	Q_0	
0	0	0	0	← Nuevo ciclo
.	.	.	.	
.	.	.	.	
.	.	.	.	
1	0	1	1	← (Nuevo ciclo)
1	1	0	0	← Siguiente estado normal

Observe que, en cualquier caso, tanto Q_0 como Q_1 toman el valor 0, mientras que a Q_2 y Q_3 se les debe forzar para que tomen el valor 0 en el duodécimo impulso de reloj. La Figura 8.7(a) muestra el contador de módulo 12. La puerta NAND decodifica parcialmente el número doce (1100) y pone a cero los flip-flops 2 y 3. Por tanto, en el duodécimo impulso de reloj, se fuerza al contador a iniciar un nuevo ciclo, pasando de once a cero, como se muestra en el diagrama de tiempos de la Figura 8.7(b). Permanece en el número doce sólo durante unos cuantos nanosegundos antes de ponerse a cero por el *glitch* en ($\overline{CLR}$).

Problema relacionado. ¿Cómo se puede modificar el contador de la Figura 8.7(a) para hacer de él un contador de módulo 13?

CONTADOR BINARIO ASÍNCRONO DE 4 BITS 74LS93



El 74LS93 es un ejemplo de circuito integrado contador asíncrono. Como muestra el diagrama lógico de la Figura 8.8, este dispositivo esta formado por un flip-flop y un contador asíncrono de 3 bits. Esta disposición le proporciona una gran flexibilidad. Si se utiliza únicamente el flip-flop, se puede utilizar como dispositivo divisor por 2; y si se utiliza únicamente el contador de 3 bits, se puede emplear como contador de módulo 8. Este dispositivo proporciona además entradas de puesta a cero (RESET) *RO(1)* y *RO(2).* Cuando estas dos entradas están a nivel ALTO, el contador se resetea al estado 0000 mediante $\overline{CLR}$.

Adicionalmente, el 74LS93A se puede utilizar como contador de 4 bits de módulo 16 (cuenta de cero a 15), conectando la salida Q_0 a la entrada CLK B, como muestra la Figura 8.9(a). También se puede configurar como contador de décadas (cuenta de 0 a 9) con reinicialización asíncrona, utilizando las entradas de puesta a cero para decodificar parcialmente el número diez, como muestra la Figura 8.9(b).

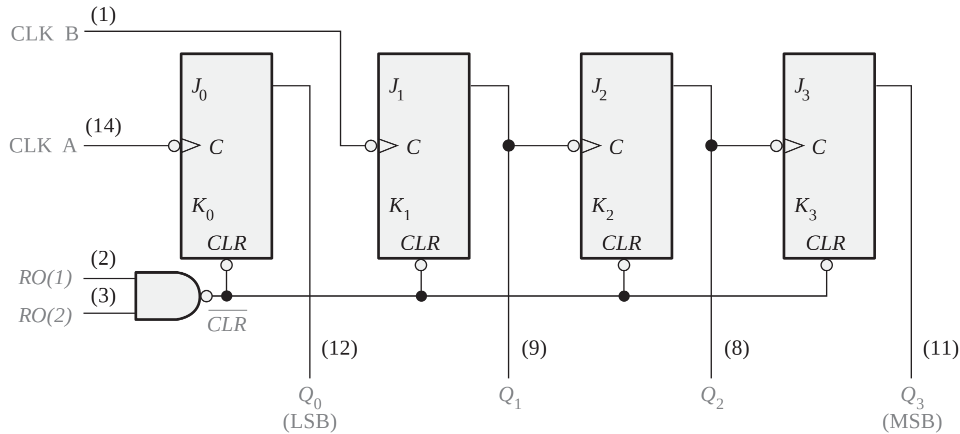
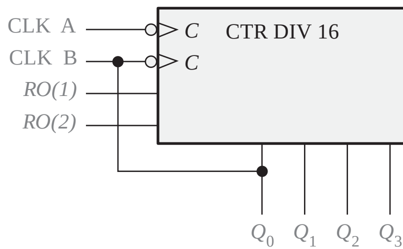
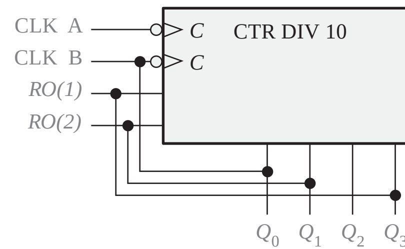


FIGURA 8.8 Diagrama lógico del contador binario asíncrono de 4 bits 74LS93. Los números de los pines se indican entre paréntesis. Todas las entradas *J* y *K* están internamente conectadas a nivel ALTO.



(a) 74LS93 conectado como un contador de módulo 16



(b) 74LS93 conectado como un contador de décadas

FIGURA 8.9 Dos configuraciones del contador asíncrono 74LS93. La etiqueta CTR DIV *n* indica un contador de *n* estados.

EJEMPLO 8.3

Explicar cómo se puede usar un 74LS93A como contador de módulo 12.

Solución Utilizar las entradas de puesta a cero *RO(1)* y *RO(2)*, para decodificar parcialmente el número 12 (recuerde que hay una puerta NAND interna asociada a estas entradas). La decodificación del número 12 se lleva cabo conectando Q_3 a *RO(1)* y Q_2 a *RO(2)*, como se muestra en la Figura 8.10. La salida Q_0 se conecta a CLK B para conseguir un contador de 4 bits.

Inmediatamente después de que el contador alcanza el estado 12 (1100), vuelve al estado inicial 0000. El inicio de un nuevo ciclo, sin embargo, origina un *glitch* en Q_2 debido a que el contador tiene que permanecer en el estado 1100 durante unos pocos nanosegundos antes de comenzar otro ciclo.

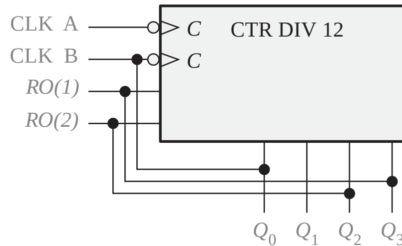


FIGURA 8.10 74LS93 conectado como contador de módulo 12.

Problema relacionado Explicar cómo se podría conectar un 74LS93 como contador de módulo 13.

REVISIÓN DE LA SECCIÓN 8.1

Las respuestas se encuentran al final del capítulo

1. ¿Qué significa el término *asíncrono* en relación a los contadores?
2. ¿Cuántos estados tiene un contador de módulo 14? ¿Cuál es el mínimo número de flip-flops requerido?

8.2 FUNCIONAMIENTO DEL CONTADOR SÍNCRONO

El término ***síncrono*** se refiere a los eventos que tienen una relación temporal fija entre sí. Un **contador síncrono** es aquel en el que todos los flip-flops del contador reciben en el mismo instante la señal de reloj.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Describir el funcionamiento de un contador síncrono binario de 2 bits. ■ Describir el funcionamiento de un contador síncrono binario de 3 bits. ■ Describir el funcionamiento de un contador síncrono binario de 4 bits. ■ Describir el funcionamiento de un contador de décadas síncrono. ■ Desarrollar los diagramas de tiempos de los contadores. ■ Utilizar el contador binario de 4 bits 74HC163 y el contador BCD de décadas 74F162.

Contador binario síncrono de 2 bits

La Figura 8.11 muestra un contador binario síncrono de 2 bits. Observe que debe utilizarse una disposición distinta a la del contador asíncrono para las entradas J_1 y K_1 de FF1, con el fin de poder conseguir una secuencia binaria.

▲ *En un contador síncrono, la entrada de reloj llega a cada flip-flop.*

El funcionamiento de este **contador síncrono** es el siguiente: en primer lugar, se supone que el contador se encuentra inicialmente en el estado binario 0; es decir, los dos flip-flops se encuentran en estado RESET. Cuando se aplica el flanco positivo del primer impulso de reloj, FF0 bascula, por lo que Q_0 se pone a nivel ALTO. ¿Qué le ocurre a FF1 en el flanco positivo de CLK1? Para averiguarlo, vamos a fijarnos en las condiciones de entrada de FF1. Las entradas J_1 y K_1 están ambas a nivel BAJO, ya que están conectadas a Q_0, y ésta todavía no se ha puesto a nivel ALTO. Recuerde que existe un retardo de propagación desde

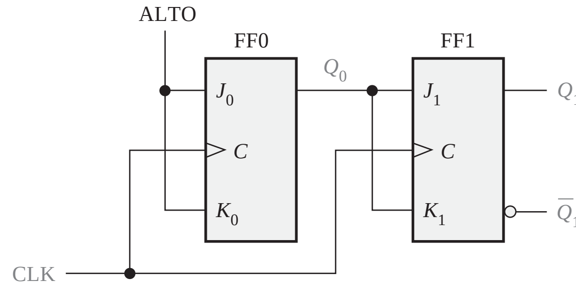


FIGURA 8.11 Contador binario síncrono de dos bits.

el flanco de disparo del impulso de reloj hasta que, realmente, se realiza la transición en la salida Q. Por tanto, $J = 0$ y $K = 0$ cuando se aplica el flanco anterior del primer impulso de reloj. Ésta es una condición de no cambio y, por tanto, FF1 no cambia de estado. En la Figura 8.12(a) se muestra una parte del diagrama de tiempos de esta fase del funcionamiento del contador.

Después de CLK1, $Q_0 = 1$ y $Q_1 = 0$ (que corresponde al estado binario 1). Cuando se produce el flanco anterior de CLK2, FF0 bascula y Q_0 se pone a nivel BAJO. Puesto que FF1 tiene un nivel ALTO ($Q_0 = 1$) en sus entradas J_1 y K_1 durante el flanco de disparo del impulso de reloj, el flip-flop bascula y Q_1 pasa a nivel ALTO. Por `tanto, después de CLK2, $Q_0 = 0$ y $Q_1 = 1$ (que corresponde al estado binario 2). En la Figura 8.12(b) se muestra en detalle esta parte del diagrama de tiempos para esta condición.

Cuando se produce el flanco anterior de CLK3, FF0 bascula de nuevo al estado SET ($Q_0 = 1$) y FF1 permanece en estado SET ($Q_1 = 1$), ya que sus entradas J_1 y K_1 están ambas a nivel BAJO ($Q_0 = 0$). Tras este flanco de disparo, $Q_0 = 1$ y $Q_1 = 1$ (que corresponde al estado binario 3). En la Figura 8.12(c) se muestra en detalle el diagrama de tiempos para esta condición.

Finalmente, durante el flanco anterior de CLK4, Q_0 y Q_1 se ponen a nivel BAJO, dado que ambos flip-flops están en modo de basculación debido al valor presente en sus entradas J y K. En la Figura 8.12(d) se muestra en detalle el diagrama de tiempos para esta condición. El contador acaba de iniciar un nuevo ciclo a partir de su estado original, 0 binario.

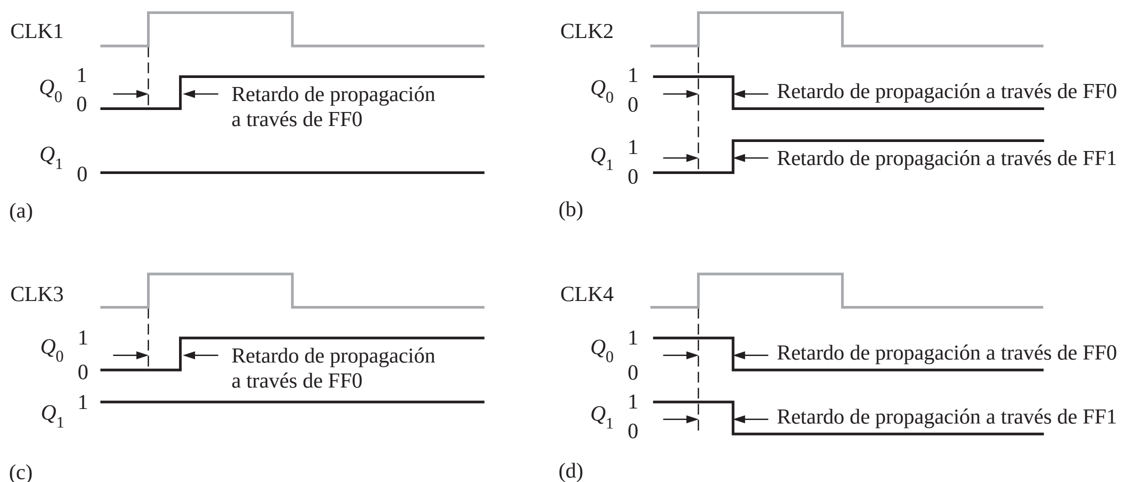


FIGURA 8.12 Diagramas de tiempos para un contador síncrono de 2 bits (los retardos de propagación de ambos flip-flops se consideran iguales).

El diagrama de tiempos completo del contador de la Figura 8.11 se muestra en la Figura 8.13. Observe que todas las transiciones de las señales son coincidentes; es decir, no se indican los retardos de propagación. Aunque los retardos son un factor importante en el funcionamiento de un contador síncrono, se suelen omitir para simplificar los diagramas de tiempos generales. Si no se muestran los pequeños retardos y las diferencias de temporización, se puede conseguir relacionar mejor las señales resultantes de un circuito lógico. Sin embargo, en circuitos digitales de alta velocidad, estos pequeños retardos son una consideración importante en el diseño y la localización de averías.

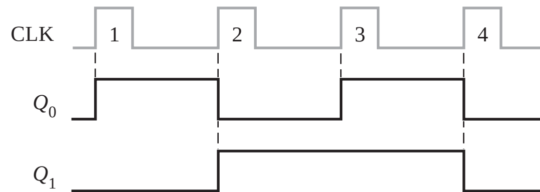


FIGURA 8.13 Diagrama de tiempos del contador de la Figura 8.11.

Contador síncrono binario de 3 bits

En la Figura 8.14 se muestra un contador síncrono binario de 3 bits y en la Figura 8.15 su diagrama de tiempos. Para entender el funcionamiento de este tipo de contador debe examinarse detenidamente su secuencia de estados, la cual se muestra en la Tabla 8.3.

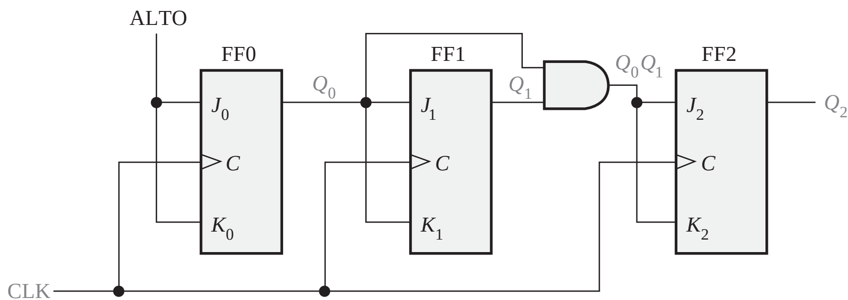


FIGURA 8.14 Contador binario síncrono de 3 bits.

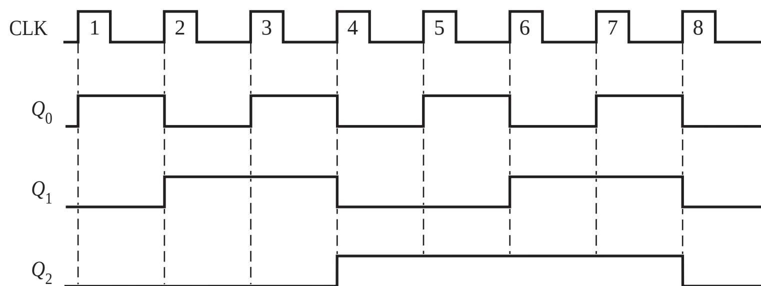


FIGURA 8.15 Diagrama de tiempos del contador de la Figura 8.14.

En primer lugar, vamos a fijarnos en Q_0. Observe que, Q_0 cambia en cada impulso de reloj a medida que el contador avanza desde su estado original hasta su estado final, para luego iniciar un nuevo ciclo a partir del

Impulso de reloj	Q_2	Q_1	Q_0
Inicialmente	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1
8 (nuevo ciclo)	0	0	0

TABLA 8.3 Secuencia de estados del contador binario de tres bits.



NOTAS INFORMÁTICAS

El TSC (*Time Stamp Counter*, contador de marca temporal) en el Pentium se utiliza para monitorizar el funcionamiento, lo que permite determinar de forma exacta una serie de importantes parámetros dentro del funcionamiento global de un sistema Pentium. Leyendo el TSC antes y después de la ejecución de un procedimiento, se puede determinar el tiempo preciso requerido para el procedimiento, en función de la frecuencia de ciclo del procesador. De esta forma, el TSC es la base para determinar todas las temporizaciones relacionadas con la optimización del funcionamiento del sistema. Por ejemplo, se puede determinar de forma precisa cuál de dos o más secuencias de programación es la más eficiente. Ésta es una herramienta muy útil para los desarrolladores de compiladores y programadores de sistemas, a la hora de generar el código más eficiente para el Pentium.

estado original. Para conseguir este funcionamiento, FF0 tiene que mantenerse en modo de basculación, aplicando constantemente niveles altos en sus entradas J_0 y K_0. Téngase en cuenta que Q_1 pasa al estado contrario cada vez que Q_0 está a 1. Este cambio se produce en CLK2, CLK4, CLK6 y CLK8. El impulso CLK8 hace que el contador inicie un nuevo ciclo. Para conseguir este modo de operación, se conecta Q_0 a las entradas J_1 y K_1 de FF1. Cuando Q_0 está a 1 y se produce un impulso de reloj, FF1 se encuentra en modo de basculación y, por tanto, cambia de estado. El resto de las veces, cuando Q_0 es 0, FF1 está en modo no cambio, quedando en su estado actual.

A continuación, vamos a ver cómo se consigue que FF2 cambie de estado en los instantes adecuados de acuerdo a la secuencia binaria. Observe que las dos veces que Q_2 cambia de estado, debe cumplirse la única condición de que tanto Q_0 como Q_1 estén a nivel ALTO. Esta condición se detecta mediante la puerta AND, cuya salida se aplica a las entradas J_2 y K_2 de FF2. Siempre que Q_0 y Q_1 están a nivel ALTO, la salida de la puerta AND hace que las entradas J_2 y K_2 de FF2 se pongan a nivel ALTO, y FF2 bascula en el siguiente impulso de reloj. El resto de las veces, las entradas J_2 y K_2 de FF2 se mantienen a nivel BAJO, al igual que la salida de la puerta AND, y FF2 no cambia de estado.

Contador síncrono binario de 4 bits

La Figura 8.16(a) presenta un contador binario síncrono de 4 bits y la Figura 8.16(b) muestra su diagrama de tiempos. Este contador particular se implementa con flip-flops disparados por flanco negativo. El razonamiento para controlar las entradas J y K de los tres primeros flip-flops es el mismo que el del contador de 3 bits, previamente estudiado. La cuarta etapa, FF3, varía sólo dos veces en la secuencia. Observe que estas dos transiciones ocurren justo cuando Q_0, Q_1 y Q_2 están a nivel ALTO. Esta condición se decodifica mediante la puer-

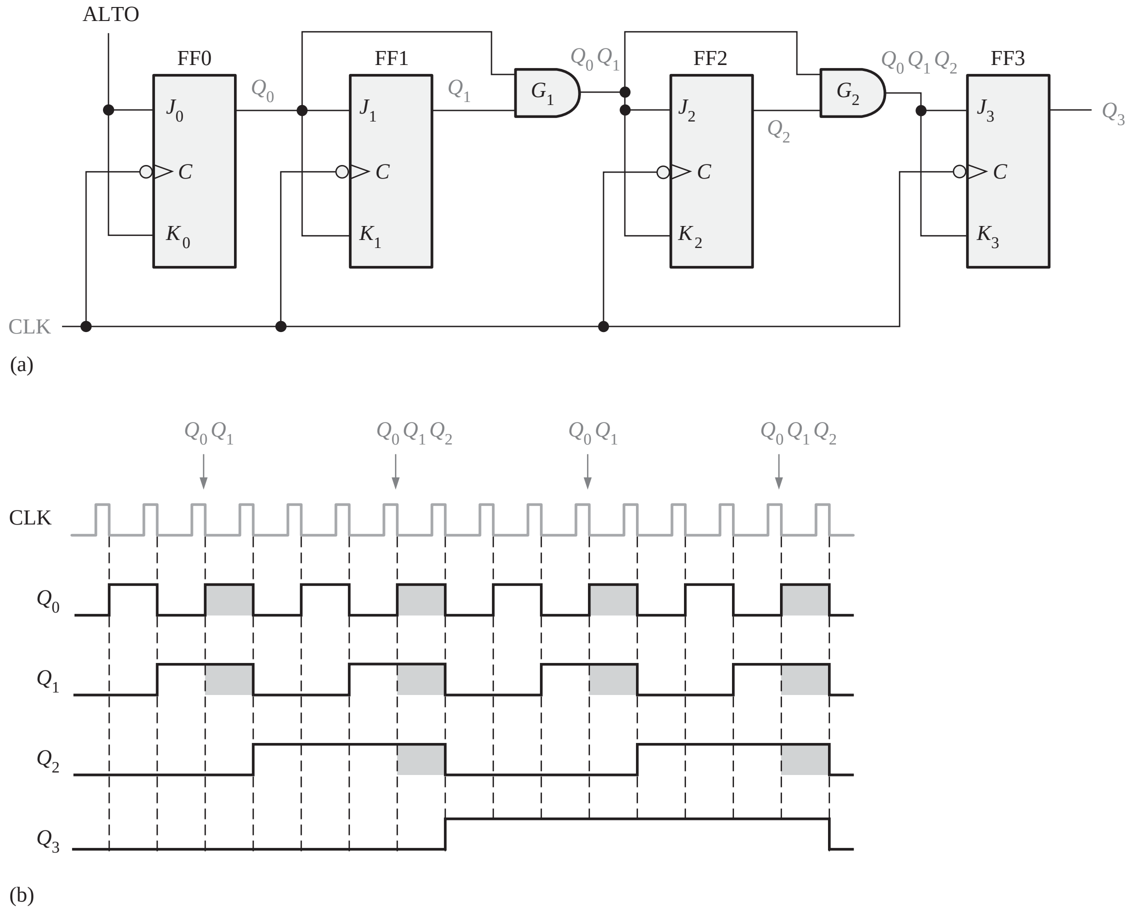


FIGURA 8.16 Contador binario síncrono de 4 bits y diagrama de tiempos. Los instantes en que las salidas de las puertas AND están a nivel ALTO se indican con áreas sombreadas.

ta AND G_2 de forma que, cuando se produce un impulso de reloj, FF3 cambia de estado. En los demás casos, las entradas J_3 y K_3 de FF3 están a nivel BAJO y se produce la condición de no cambio.

Contador de décadas síncrono de 4 bits

Como ya sabemos, un contador de décadas BCD dispone de una secuencia binaria truncada que va desde 0000 hasta el estado 1001. En lugar de pasar al estado 1010, inicia un nuevo ciclo a partir del estado 0000. En la Figura 8.17 se presenta un contador de décadas BCD síncrono. En la Figura 8.18 se muestra el diagrama de tiempos para este contador de décadas.

El funcionamiento de este contador se puede entender examinando la secuencia de estados de la Tabla 8.4, y siguiendo la implementación de la Figura 8.17. En primer lugar, observe que FF0 (Q_0) bascula en cada impulso de reloj, por lo que la ecuación lógica para sus entradas J_0 y K_0 es:

$$J_0 = K_0 = 1$$

Esta ecuación se implementa conectando J_0 y K_0 a un nivel ALTO constante.

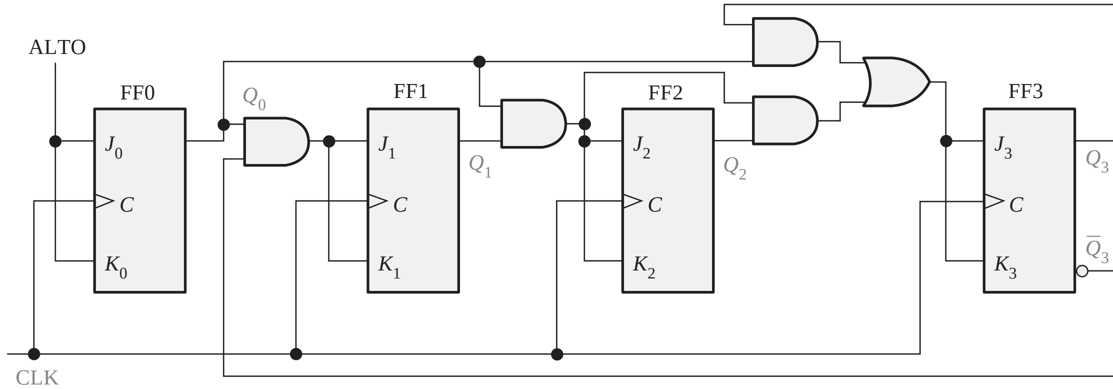


FIGURA 8.17 Contador de décadas BCD síncrono.

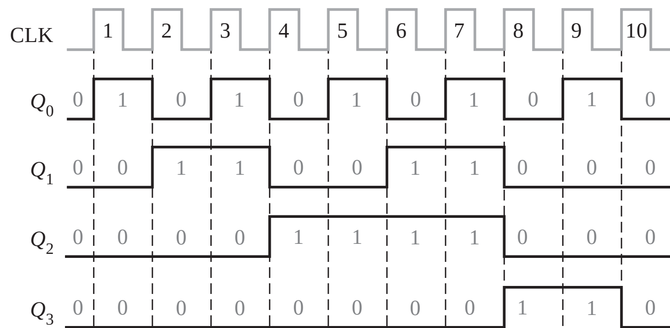


FIGURA 8.18 Diagrama de tiempos del contador de décadas BCD (Q_0 es el LSB).

A continuación, observe en la Tabla 8.4 que FF1 (Q_1) cambia en el siguiente impulso de reloj cada vez que $Q_0 = 1$ y $Q_3 = 0$, por lo que la ecuación lógica para las entradas J_1 y K_1 es:

$$J_1 = K_1 = Q_0\overline{Q}_3$$

Esta ecuación se implementa aplicando la operación AND a las salidas Q_0 y $\overline{Q}_3$, y conectando la salida de la puerta a las entradas J_1 y K_1 de FF1.

El flip-flop 2 (Q_2) cambia de estado en el siguiente impulso de reloj cada vez que $Q_0 = 1$ y $Q_1 = 1$. Luego la ecuación lógica de entrada es:

$$J_2 = K_2 = Q_0Q_1$$

Esta ecuación se implementa aplicando Q_0 y Q_1 a las entradas de una puerta AND, y conectando la salida de la puerta a las entradas J_2 y K_2 de FF2.

Finalmente, FF3 (Q_3) cambia de estado en el siguiente impulso de reloj cada vez que $Q_0 = 1$, $Q_1 = 1$ y $Q_2 = 1$ (estado 7), o cuando $Q_0 = 1$ y $Q_3 = 1$ (estado 9). La ecuación que rige esto es la siguiente:

$$J_3 = K_3 = Q_0Q_1Q_2 + Q_0Q_3$$

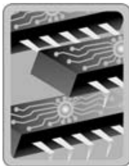
Esta función se implementa mediante la lógica AND/OR conectada a las entradas J_3 y K_3 de FF3, como se muestra en el diagrama lógico de la Figura 8.17. Observe que la única diferencia entre este contador de déca-

Impulso de reloj	Q_3	Q_2	Q_1	Q_0
Inicialmente	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10 (nuevo ciclo)	0	0	0	0

TABLA 8.4 Estados del contador de décadas BCD.

das y el contador binario de módulo 16 de la Figura 8.16 son la puerta AND con entradas $Q_0\overline{Q}_3$, la puerta AND con entradas Q_0Q_3 y la puerta OR; esta disposición detecta la ocurrencia del estado 1001 y hace que el contador inicie un nuevo ciclo correctamente en el siguiente impulso de reloj.

CONTADOR BINARIO SÍNCRONO DE 4 BITS 74HC163



El 74HC163 es un ejemplo de un circuito integrado contador binario síncrono de 4 bits. El símbolo lógico se muestra en la Figura 8.19, con la numeración de pines entre paréntesis. Este contador tiene varias características adicionales con respecto a las características básicas del contador binario síncrono general previamente tratado.

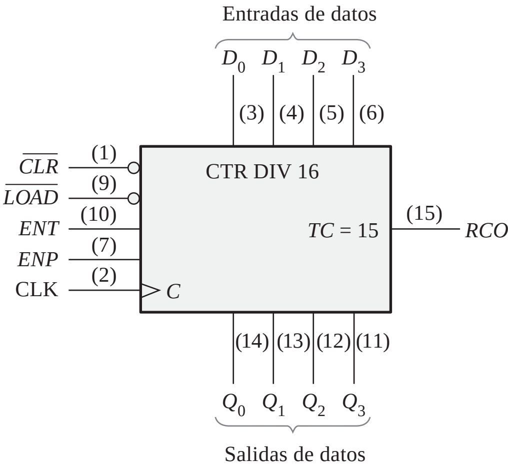


FIGURA 8.19 El contador binario síncrono de 4 bits 74HC163. La etiqueta CTR DIV 16 indica un contador con dieciséis estados.

En primer lugar, el contador puede reinicializarse de forma síncrona en cualquier número binario de 4 bits, aplicando los niveles adecuados en las entradas de datos paralelo. Cuando se aplica un nivel BAJO a la entrada $\overline{LOAD}$, el contador asumirá el estado de las entradas de datos en el siguiente impulso de reloj. Por tanto, la secuencia del contador se puede iniciar con cualquier número binario de 4 bits.

Además, hay una entrada de borrado activa a nivel BAJO ($\overline{CLR}$) que pone a cero de forma síncrona los cuatro flip-flops del contador. Hay dos entradas de habilitación, *ENP* y *ENT*. Estas entradas deben estar a nivel ALTO para que el contador pueda avanzar a través de su secuencia de estados binarios. Cuando al menos una de las entradas está a nivel BAJO, el contador se desactiva. La salida de propagación de reloj (*Ripple Clock Output*, *RCO*) se pone a nivel ALTO cuando el contador alcanza el ***valor de fin de cuenta*** (*Terminal Count*, *TC*) de quince (*TC* = 15). Esta salida, junto con las entradas de habilitación permiten que estos contadores se puedan disponer en cascada para conseguir secuencias de cuenta mayores.

La Figura 8.20 muestra un diagrama de tiempos de este contador, que se inicializa en el estado 12 (1100) y luego avanza hasta su valor de fin de cuenta 15 (1111). La entrada D_0 corresponde al bit de entrada menos significativo y Q_0 es el bit de salida menos significativo.

Vamos a examinar este diagrama de tiempos en detalle. Esto nos ayudará a interpretar los diagramas de tiempos que encontraremos más adelante en este mismo capítulo o

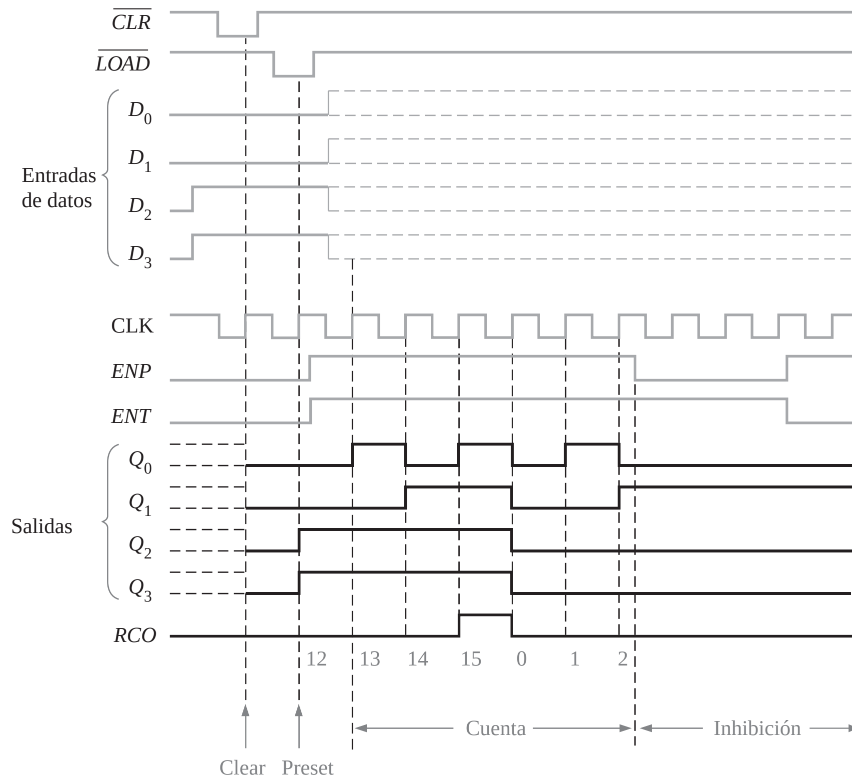


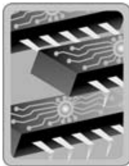
FIGURA 8.20 Ejemplo de un diagrama de tiempos para un 74HC163.

en las hojas de características de los fabricantes. Para comenzar, el impulso a nivel BAJO en la entrada $\overline{CLR}$ hace que todas las salidas (Q_0, Q_1, Q_2 y Q_3) se pongan a nivel BAJO.

A continuación, el impulso a nivel BAJO en la entrada $\overline{LOAD}$ introduce de forma síncrona los datos en las entradas (D_0, D_1, D_2 y D_3) del contador. Éstos aparecen en las salidas Q cuando se produce el primer flanco positivo de reloj después de que $\overline{LOAD}$ pasa a nivel BAJO. Ésta es la operación de inicialización (PRESET). En este ejemplo particular, Q_0 está a nivel BAJO, Q_1 está a nivel BAJO, Q_2 está a nivel ALTO y Q_3 está a nivel ALTO. Por supuesto, esto corresponde al número decimal 12 (Q_0 es el bit menos significativo).

Ahora, el contador avanza por los estados 13, 14 y 15 en los tres siguientes flancos positivos de reloj, y luego comienza un nuevo ciclo en 0, 1, 2 con los siguientes impulsos de reloj. Observe que las dos entradas *ENP* y *ENT* están a nivel ALTO durante la secuencia de estados. Cuando *ENP* pasa a nivel BAJO, el contador se inhibe y permanece en el estado binario 2.

CONTADOR DE DÉCADAS BCD SÍNCRONO 74F162



El 74F162 es un ejemplo de un contador de décadas. Se puede inicializar con cualquier número BCD utilizando las entradas de datos con la entrada $\overline{PE}$ a nivel BAJO. Un nivel BAJO en la entrada $\overline{SR}$ asíncrona pone en estado RESET al contador. Las entradas de habilitación *CEP* y *CET* tienen que estar ambas a nivel ALTO para que el contador avance a través de la secuencia de estados, en respuesta a una transición positiva en la entrada de reloj CLK. Las entradas de habilitación junto con el valor de fin de cuenta, *TC* (1001), permiten conectar varios contadores de décadas en cascada. La Figura 8.21 muestra el símbolo lógico del contador 74F162 y la Figura 8.22 presenta un diagrama de tiempos del contador inicializado en el estado 7 (0111). Los contadores en cascada se tratarán en la Sección 8.5.

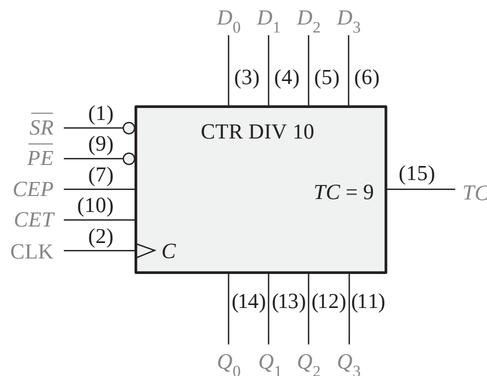


FIGURA 8.21 El contador de décadas BCD síncrono 74F162. La etiqueta CTR DIV 10 indica un contador con diez estados.

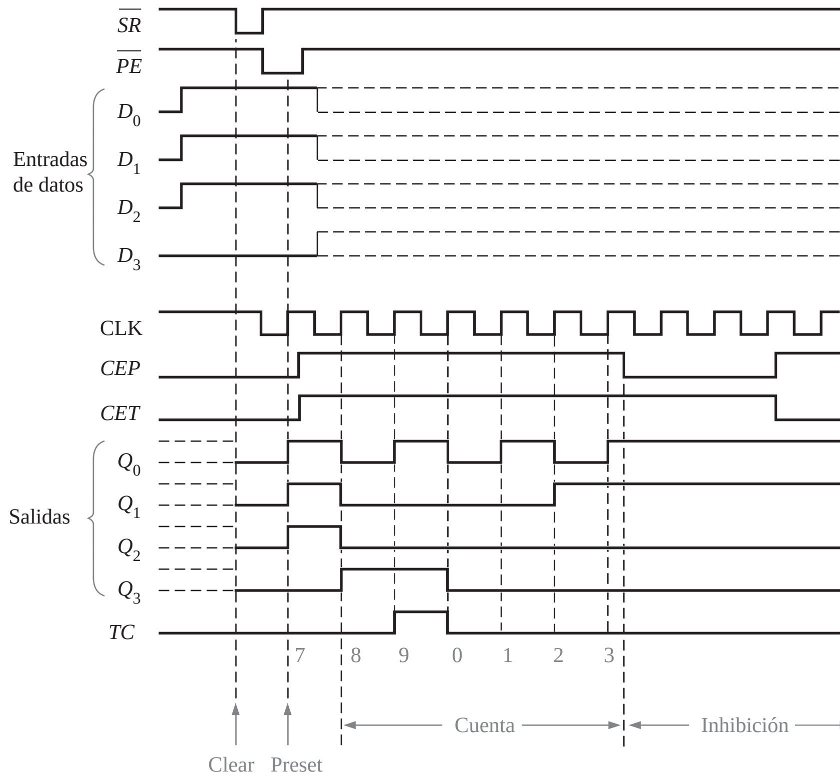


FIGURA 8.22 Ejemplo de diagrama de tiempos de un 74F162.

REVISIÓN DE LA SECCIÓN 8.2

1. ¿En qué se diferencia un contador síncrono de uno asíncrono?
2. Explicar el funcionamiento de la función preset de contadores tales como el 74HC163.
3. Describir el propósito de las entradas *ENP* y *ENT*, y de la salida *RCO* para el contador 74HC163.

8.3 CONTADOR SÍNCRONO ASCENDENTE/DESCENDENTE

Un **contador ascendente/descendente** (up/down) es aquel capaz de progresar en cualquier dirección a lo largo de una cierta secuencia. Un contador ascendente/descendente, algunas veces también denominado contador bidireccional, puede tener cualquier secuencia de estados especificada. Un contador binario de 3 bits que avanza en modo ascendente a través de la secuencia (0, 1, 2, 3, 4, 5, 6, 7) y que luego puede invertirse para recorrer la secuencia en sentido contrario (7, 6, 5, 4, 3, 2, 1, 0) es un ejemplo de un modo de operación secuencial ascendente/descendente.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Explicar el funcionamiento básico de un contador ascendente/descendente. ■ Utilizar el contador de décadas ascendente/descendente 74HC190.

En general, la mayoría de los contadores ascendentes/descendentes pueden invertirse en cualquier punto de su secuencia. Por ejemplo, el contador binario de 3 bits se puede configurar para que realice la siguiente secuencia:

$$\overbrace{0,1,2,3,4,5}^{\text{ASCENDENTE}},\ \underbrace{4,3,2}_{\text{DESCENDENTE}},\ \overbrace{3,4,5,6,7}^{\text{ASCENDENTE}},\ \underbrace{6,5}_{\text{DESCENDENTE}},\ \text{etc.}$$

La Tabla 8.5 muestra la secuencia ascendente/descendente (up/down) completa de un contador binario de 3 bits. Las flechas indican los movimientos entre los estados del contador, tanto para el modo ASCENDENTE como para el modo Descendente. Un examen de Q_0 para ambas secuencias, ascendente y descendente, muestra que FF0 bascula con cada impulso de reloj. Luego las entradas J_0 y K_0 de FF0 son:

$$J_0 = K_0 = 1$$

Para la secuencia ascendente, Q_1 cambia de estado en el siguiente impulso de reloj cuando $Q_0 = 1$. Para la secuencia descendente, Q_1 cambia en el siguiente impulso de reloj cuando $Q_0 = 0$. Por tanto, las entradas J_1 y K_1 de FF1 tienen que ser igual a 1, para las condiciones expresadas en la siguiente ecuación:

$$J_1 = K_1 = (Q_0 \cdot \text{UP}) + (\overline{Q}_0 \cdot \text{DOWN})$$

Para la secuencia ascendente, Q_2 cambia de estado en el siguiente impulso de reloj cuando $Q_0 = Q_1 = 1$. Para la secuencia descendente, Q_2 cambia en el siguiente impulso de reloj cuando $Q_0 = Q_1 = 0$. Por tanto, las entradas J_2 y K_2 de FF2 tienen que ser igual a 1, para las condiciones expresadas en la siguiente ecuación:

$$J_2 = K_2 = (Q_0 \cdot Q_1 \cdot \text{UP}) + (\overline{Q}_0 \cdot \overline{Q}_1 \cdot \text{DOWN})$$

Cada una de las condiciones para las entradas *J* y *K* de cada flip-flop produce una basculación en el punto apropiado de la secuencia del contador.

La Figura 8.23 muestra una implementación básica de un contador binario de 3 bits ascendente/ descendente, utilizando las ecuaciones lógicas que acabamos de desarrollar para las entradas *J* y *K* de cada flip-flop. Observe que, la entrada de control $UP/\overline{DOWN}$ ($ascendente/\overline{descendente}$) está a nivel ALTO cuando trabaja en modo ascendente y a nivel BAJO cuando trabaja en modo descendente.

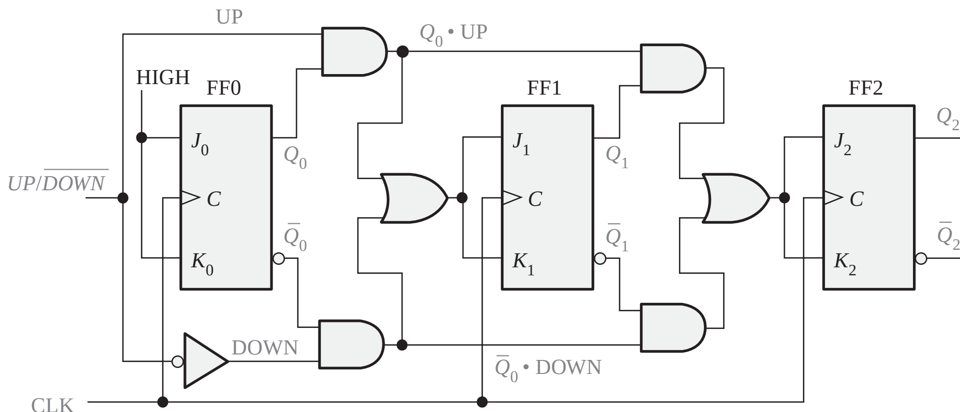


FIGURA 8.23 Contador síncrono ascendente/descendente básico de 3 bits.

Impulso de reloj	ASC.	Q_2	Q_1	Q_0	DESC.
0		0	0	0	
1		0	0	1	
2		0	1	0	
3		0	1	1	
4		1	0	0	
5		1	0	1	
6		1	1	0	
7		1	1	1	

TABLA 8.5 Secuencia ascendente/descendente de un contador binario de 3 bits.

EJEMPLO 8.4

Dibujar el diagrama de tiempos y determinar la secuencia de un contador síncrono binario de 4 bits ascendente/descendente, si el reloj y las entradas de control $UP/\overline{DOWN}$ son las señales que se muestran en la Figura 8.24(a). El contador se inicializa en el estado cero y es de tipo disparado por flanco positivo.

FIGURA 8.24

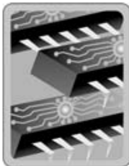
Solución En la Figura 8.24(b) se presenta el diagrama de tiempos, mostrando las salidas *Q*. A partir de estas formas de onda, la secuencia del contador es la que se indica en la Tabla 8.6.

Problema relacionado Dibujar el diagrama de tiempos si se invierte la señal de control $UP/\overline{DOWN}$ de la Figura 8.24(a).

Q_3	Q_2	Q_1	Q_0	
0	0	0	0	ASCENDENTE
0	0	0	1	
0	0	1	0	
0	0	1	1	
0	1	0	0	
0	0	1	1	DESCENDENTE
0	0	1	0	
0	0	0	1	
0	0	0	0	
1	1	1	1	
0	0	0	0	ASCENDENTE
0	0	0	1	
0	0	1	0	
0	0	0	1	DESCENDENTE
0	0	0	0	

TABLA 8.6

CONTADOR DE DÉCADAS ASCENDENTE/DESCENDENTE 74HC190



La Figura 8.25 muestra el diagrama lógico del 74HC190, buen ejemplo de un circuito integrado contador ascendente/descendente. La dirección de la cuenta se determina por el nivel de la entrada up/down ($D/\overline{U}$). Cuando esta entrada está a nivel ALTO, el contador se decrementa (desciende); cuando está a nivel BAJO, el contador se incrementa (asciende). Además, este dispositivo se puede inicializar en cualquier dígito BCD que se desee, el cual se carga a través de las entradas de datos cuando la entrada $\overline{LOAD}$ está a nivel BAJO.

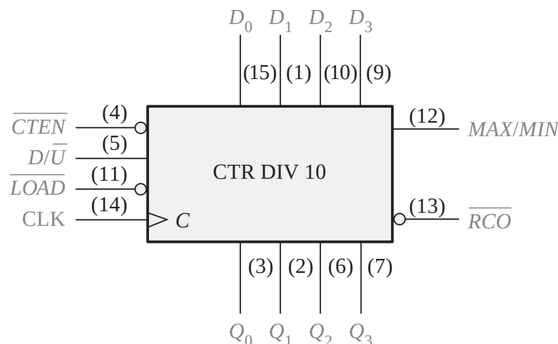


FIGURA 8.25 El contador de décadas ascendente/descendente síncrono 74HC190.

La salida *MAX/MIN* produce un impulso a nivel ALTO cuando se alcanza el valor de fin de cuenta de nueve (1001) en el modo ascendente, o cuando se alcanza el valor de fin de cuenta de cero (0000) en el modo descendente. La salida *MAX/MIN*, junto con la salida de propagación de reloj ($\overline{RCO}$) y la entrada de habilitación de cuenta ($\overline{CTEN}$), se usa para conectar contadores en cascada (los contadores en cascada se verán en la Sección 8.5).

La Figura 8.26 es un diagrama de tiempos, que muestra un contador 74HC190 inicializado en siete (0111); el contador luego describe una secuencia ascendente, seguida de una secuencia descendente. La salida *MAX/MIN* está a nivel ALTO cuando el contador está en el estado 0 (*MIN*) o en el estado 1001 (*MAX*).

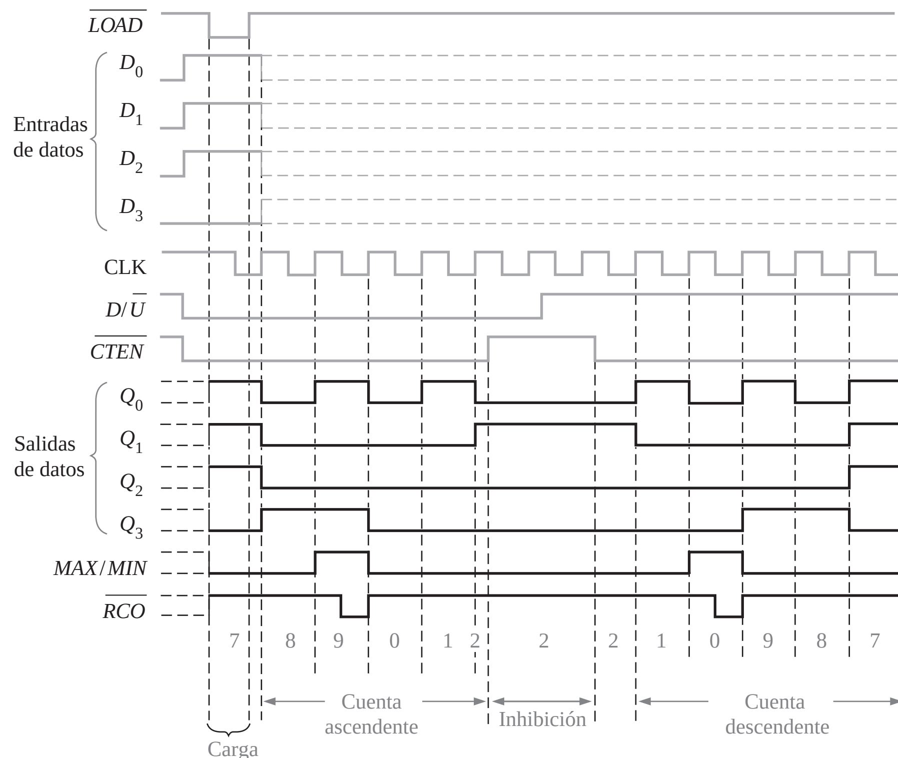


FIGURA 8.26 Ejemplo de un diagrama de tiempos para un 74HC190.

REVISIÓN DE LA SECCIÓN 8.3

1. Un contador binario de 4 bits ascendente/descendente se encuentra en modo descendente y en el estado 1010. En el siguiente impulso de reloj, ¿a qué estado pasa?

2. ¿Cuál es el valor de fin de cuenta de un contador binario de 4 bits en modo ascendente? ¿Y en modo descendente? ¿Cuál es el siguiente estado después del valor final de cuenta en modo descendente?

8.4 DISEÑO DE CONTADORES SÍNCRONOS

En esta sección veremos cómo pueden aplicarse técnicas de diseño de circuitos secuenciales específicamente al diseño de contadores. En general, los circuitos secuenciales pueden clasificarse en dos tipos; (1) aquellos en los que la salida o salidas dependen únicamente del estado interno actual (denominados *circuitos de Moore*) y (2) aquellos en los que la salida o salidas dependen tanto del estado actual como de la entrada o entradas (denominados *circuitos de Mealy*). Esta sección es opcional y puede ser omitida sin que afecte al material expuesto en lo que resta de libro. Se recomienda el estudio de esta sección a aquéllos que deseen una introducción al diseño de contadores o de máquinas de estados en general. No es necesario su conocimiento para abordar otros temas.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Describir un circuito secuencial general en función de sus partes básicas, y de sus entradas y salidas. ■ Desarrollar un diagrama de estados para una determinada secuencia. ■ Desarrollar una tabla del estado siguiente para una secuencia de contador específica. ■ Crear una tabla de transiciones de flip-flops. ■ Utilizar el método del mapa de Karnaugh para obtener los requisitos lógicos de un contador asíncrono. ■ Implementar un contador para generar una secuencia de estados específica.

Modelo general de un circuito secuencial

Antes de exponer una técnica de diseño de contadores específica, vamos a comenzar con una definición general de **circuito secuencial** o ***máquina de estados***: un circuito secuencial está formado por una etapa de lógica combinacional y una sección de memoria (flip-flops), como se muestra en la Figura 8.27. En un circuito secuencial sincronizado, hay una entrada de reloj en la etapa de memoria, tal como se indica.

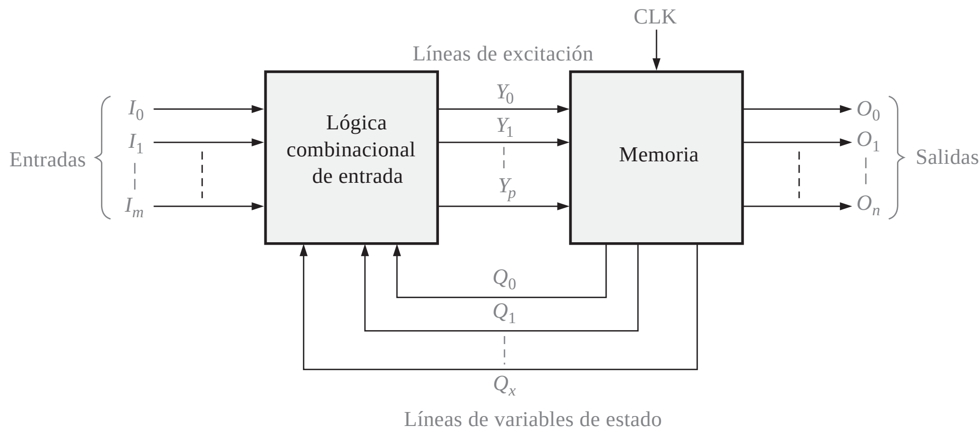


FIGURA 8.27 Circuito secuencial general sincronizado.

Para el correcto funcionamiento del circuito se requiere la información almacenada en la etapa de memoria, así como las entradas de la lógica combinacional (I_0, I_1, ..., I_m). En cualquier instante de tiempo, la memoria se encuentra en un estado denominado *estado actual* y avanza al *estado siguiente* con un impulso de reloj, determinado por las condiciones de las líneas de excitación (Y_0, Y_1, ..., Y_p). El estado actual de la memoria se representa por las variables de estado (Q_0, Q_1, ..., Q_x). Estas variables de estado, junto con las entradas (I_0, I_1, ..., I_m), determinan las salidas del sistema (O_0, O_1, ..., O_n).

No todos los circuitos secuenciales tienen variables de entrada y salida como en el modelo general que se acaba de presentar. Sin embargo, todos tienen variables de excitación y variables de estado. Los contadores son un caso particular de los circuitos secuenciales sincronizados. En esta sección, se aplica un procedimiento de diseño general de los circuitos secuenciales a los contadores síncronos a través de una serie de pasos.

Paso 1: diagrama de estados

El primer paso en el diseño de un contador consiste en crear un diagrama de estados. Un ***diagrama de estados*** muestra la progresión de estados por los que el contador avanza cuando se aplica una señal de reloj. Como ejemplo, en la Figura 8.28, se muestra un diagrama de estados de un contador básico en código Gray de 3 bits. Este circuito particular no tiene ninguna entrada aparte de la de reloj, y ninguna otra salida más que las que se toman en cada flip-flop del contador. Si lo desea, puede repasar el código Gray, descrito en el Capítulo 2.

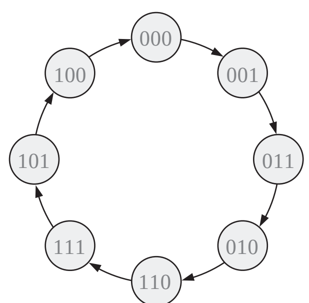


FIGURA 8.28 Diagrama de estados para un contador en código Gray de 3 bits.

Paso 2: tabla del estado siguiente

Una vez que se define el circuito secuencial mediante un diagrama de estados, el segundo paso consiste en obtener una tabla del estado siguiente, que enumera cada estado del contador (estado actual) junto con el correspondiente estado siguiente. *El estado siguiente es el estado al que el contador pasa desde su estado actual, al aplicar un impulso de reloj*. La tabla del estado siguiente se obtiene a partir del diagrama de estados, y se muestra en la Tabla 8.7 para el contador en código Gray de 3 bits. Q_0 es el bit menos significativo.

Estado actual			Estado siguiente		
Q_2	Q_1	Q_0	Q_2	Q_1	Q_0
0	0	0	0	0	1
0	0	1	0	1	1
0	1	1	0	1	0
0	1	0	1	1	0
1	1	0	1	1	1
1	1	1	1	0	1
1	0	1	1	0	0
1	0	0	0	0	0

TABLA 8.7 Tabla del estado siguiente para el contador en código Gray de 3 bits.

Paso 3: tabla de transiciones de los flip-flops

La Tabla 8.8 es una tabla de transiciones del flip-flop J-K. Se enumeran todas las posibles transiciones de salida, mostrando cómo evoluciona la salida *Q* del flip-flop al pasar de los estados actuales a los estados siguientes. Q_N es el estado presente en el flip-flop (antes de un impulso de reloj) y Q_{N+1} es el estado siguiente (después de un impulso de reloj). Para cada transición de salida, se indican las entradas *J* y *K* que dan lugar a la transición. Las "X" indican condiciones indiferentes (la entrada puede ser un 1 o un 0).

Al diseñar el contador, se aplica la tabla de transiciones a cada flip-flop del contador, la cual está basada en la tabla del estado siguiente (Tabla 8.7). Por ejemplo, para el estado actual 000, Q_0 pasa del estado actual 0 al estado siguiente 1. Para que esto ocurra, J_0 tiene que ser 1 y es indiferente el valor que tome K_0 (J_0 = 1, K_0 = X), como se indica en la tabla de transiciones (Tabla 8.8). A continuación, el estado actual de Q_1 es 0 y permanece en 0 en el estado siguiente. Para esta transición, J_1 = 0 y K_1 = X. Por último, el estado actual de Q_2 es 0 y permanece en 0 en el estado siguiente. Por tanto, J_2 = 0 y K_2 = X. Este análisis se repite para cada estado actual definido en la Tabla 8.7.

Transiciones de salida			Entradas del flip-flop	
Q_N		Q_{N+1}	*J*	*K*
0	⟶	0	0	X
0	⟶	1	1	X
1	⟶	0	X	1
1	⟶	1	X	0

Q_N: estado actual

Q_{N+1} : siguiente estado

X: condición "indiferente"

TABLA 8.8 Tabla de transiciones para un flip-flop J-K.

Paso 4: mapas de Karnaugh

Los mapas de Karnaugh se utilizan para determinar la lógica requerida para las entradas *J* y *K* de cada flip-flop del contador. Se debe utilizar un mapa de Karnaugh para la entrada *J* y otro para la entrada *K* de cada flip-flop. En este procedimiento de diseño, cada celda del mapa de Karnaugh representa uno de los estados actuales de la secuencia del contador enumerados en la Tabla 8.7.

A partir de los estados *J* y *K* de la tabla de transiciones (Tabla 8.8) se introduce un 1, un 0 o una X en cada celda de la tabla correspondiente al estado actual, dependiendo de la transición de la salida *Q* de cada flip-flop en particular. Para ilustrar este procedimiento, se muestran en la Figura 8.29 dos valores de entrada de ejemplo para las entradas J_0 y K_0 del flip-flop menos significativo (Q_0).

Los mapas de Karnaugh completos de los tres flip-flops del contador se muestran en la Figura 8.30. Las celdas se agrupan tal como se indica, obteniéndose las expresiones booleanas correspondientes para cada grupo.

Paso 5: expresiones lógicas para las entradas de los flip-flops

A partir de los mapas de Karnaugh de la Figura 8.30 se obtienen las siguientes expresiones para las entradas *J* y *K* de cada flip-flop:

$$J_0 = Q_2Q_1 + \overline{Q}_2\overline{Q}_1 = \overline{Q_2 \oplus Q_1}$$

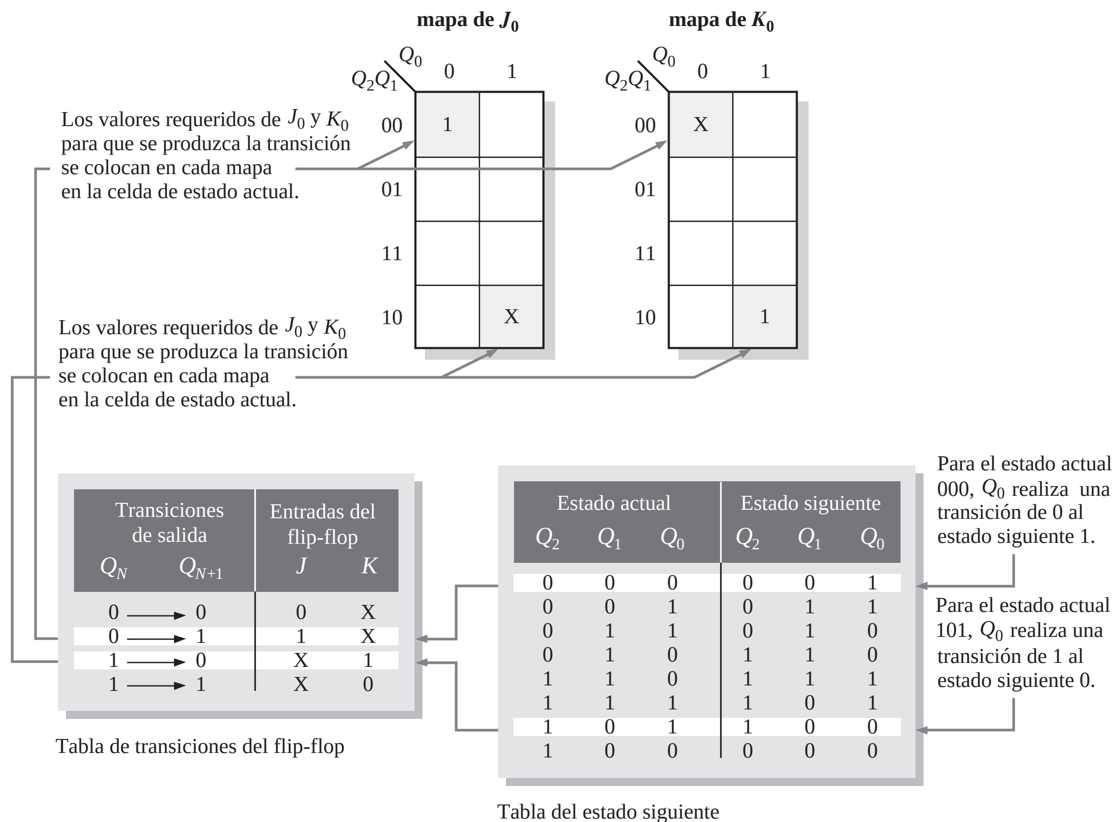


FIGURA 8.29 Ejemplos del procedimiento de utilización de mapas de Karnaugh para la secuencia de contador representada en las Tablas 8.7 y 8.8.

$$K_0 = Q_2\overline{Q}_1 + \overline{Q}_2 Q_1 = Q_2 \oplus Q_1$$

$$J_1 = \overline{Q}_2 Q_0$$

$$K_1 = Q_2 Q_0$$

$$J_2 = Q_1\overline{Q}_0$$

$$K_2 = \overline{Q}_1\overline{Q}_0$$

Paso 6: implementación del contador

El paso final consiste en implementar la lógica combinacional a partir de las expresiones de las entradas *J* y *K,* y conectar los flip-flops para conseguir un contador en código Gray de 3 bits, como se muestra en la Figura 8.31.

A continuación, se expone un resumen de los pasos dados en el diseño de este contador. En general, estos pasos se pueden aplicar a cualquier circuito secuencial.

1. Especificar la secuencia del contador y dibujar un diagrama de estados.
2. Obtener la tabla del estado siguiente a partir del diagrama de estados.

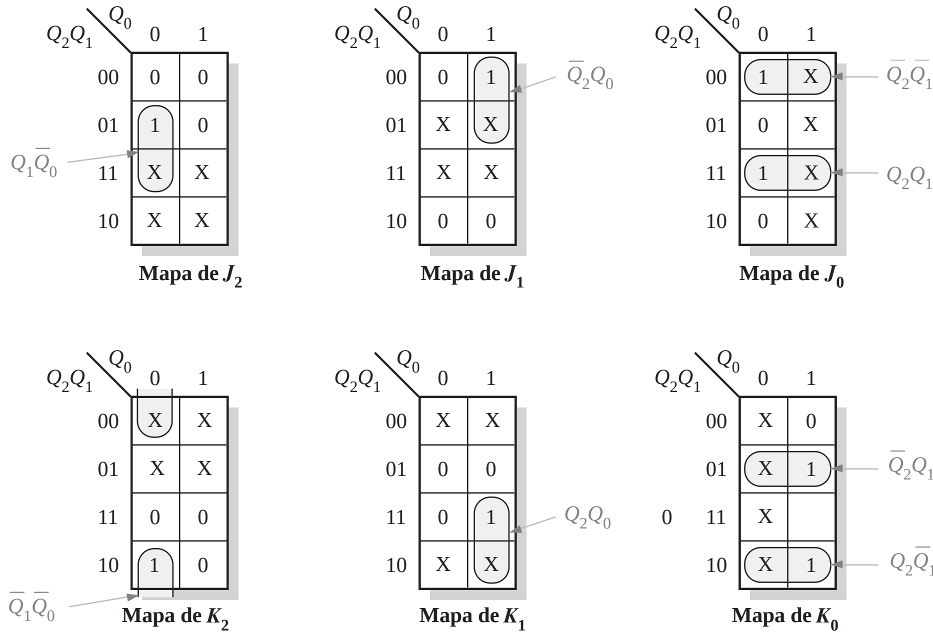


FIGURA 8.30 Mapas de Karnaugh para las entradas *J* y *K* del estado actual.

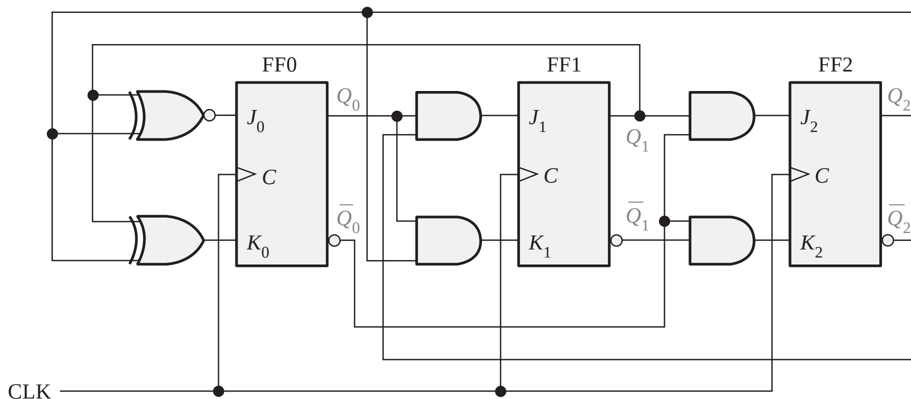


FIGURA 8.31 Contador en código de Gray de 3 bits.

3. Desarrollar una tabla de transiciones que muestre las entradas del flip-flop requeridas para cada transición. La tabla de transiciones es siempre la misma para cada tipo de flip-flop.
4. Transferir los estados *J* y *K* de la tabla de transiciones al mapa de Karnaugh. Utilizar un mapa de Karnaugh para cada entrada de cada flip-flop.
5. Formar los términos productos a partir de los mapas para generar una expresión lógica, para cada entrada de los flip-flops.
6. Implementar la expresión con lógica combinacional y conectarla a los flip-flops para crear el contador.

A continuación, en los Ejemplos 8.5 y 8.6, se va a aplicar este procedimiento al diseño de otros contadores síncronos.

EJEMPLO 8.5

Diseñar un contador que realice la secuencia de cuenta binaria irregular que se muestra en el diagrama de estados de la Figura 8.32. Utilizar flip-flops J-K.

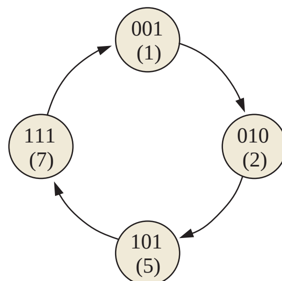


FIGURA 8.32

Solución **Paso 1.** El diagrama de estados es el que se muestra en la figura. Aunque hay sólo cuatro estados, necesitamos un contador de 3 bits para implementar esta secuencia, dado que el número binario máximo es siete. Ya que la secuencia requerida no incluye todos los posibles estados binarios, los estados no válidos (0, 3, 4 y 6) pueden ser considerados como indiferentes en el diseño. Sin embargo, si el contador pasara por error por un estado no válido, debe asegurarse que luego volverá a un estado válido.

Paso 2. La tabla del estado siguiente se desarrolla a partir del diagrama de estados y se muestra en la Tabla 8.9.

Estado actual			Estado siguiente		
Q_2	Q_1	Q_0	Q_2	Q_1	Q_0
0	0	1	0	1	0
0	1	0	1	0	1
1	0	1	1	1	1
1	1	1	0	0	1

TABLA 8.9 Tabla del estado siguiente.

Transiciones de salida			Entradas del flip-flop	
Q_N		Q_{N+1}	J	K
0	⟶	0	0	X
0	⟶	1	1	X
1	⟶	0	X	1
1	⟶	1	X	0

TABLA 8.10 Tabla de transiciones para un flip-flop J-K.

Paso 3. En la Tabla 8.10 se repite la tabla de transiciones del flip-flop J-K.

Paso 4. En la Figura 8.33 se muestran las entradas J y K en los mapas de Karnaugh del estado actual. También se pueden incluir condiciones indiferentes (X) en las celdas correspondientes a los estados no válidos 000, 011, 100 y 110.

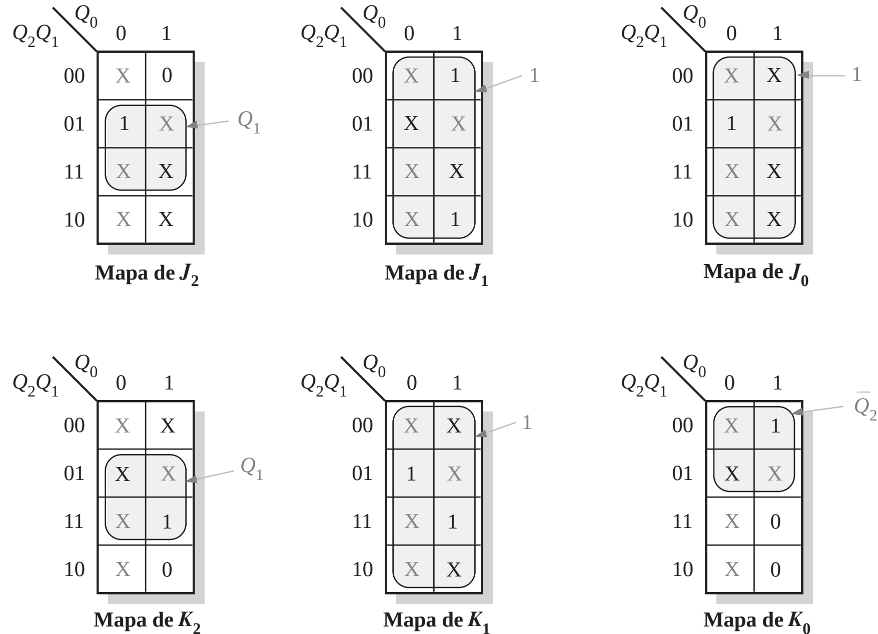


FIGURA 8.33

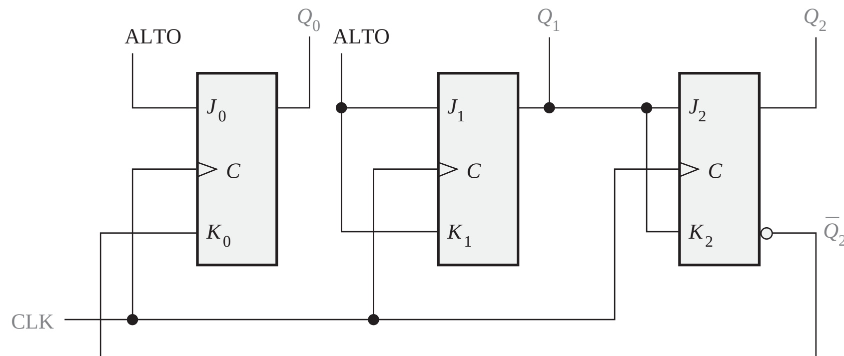


FIGURA 8.34

Paso 5. Agrupar los 1s, utilizando tantos estados indiferentes como sea posible para obtener la máxima simplificación, como se muestra en la Figura 8.33. Observe que cuando se agrupan *todas* las celdas de una tabla, la expresión es simplemente 1. La expresión para cada entrada J y K a partir de los mapas es la que sigue:

$$J_0 = 1, K_0 = \overline{Q}_2$$
$$J_1 = K_1 = 1$$
$$J_2 = K_2 = Q_1$$

Paso 6. La implementación del contador se puede ver en la Figura 8.34.

Un análisis demuestra que si el contador, por accidente, cae en uno de los estados no válidos (0, 3, 4, 6) devolverá siempre un estado válido de acuerdo con las siguientes secuencias $0 \rightarrow 3 \rightarrow 4 \rightarrow 7$ y $6 \rightarrow 1$.

Problema relacionado Verificar el análisis que demuestra que el contador (al final) siempre va a pasar a un estado válido desde un estado no válido.

EJEMPLO 8.6

Desarrollar un contador síncrono ascendente/descendente de 3 bits con una secuencia en código Gray. El contador trabajará en modo ascendente cuando la entrada de control $UP/\overline{DOWN}$ sea 1, y trabajará en modo descendente cuando la entrada de control sea 0.

Solución **Paso 1.** El diagrama de estados se muestra en la Figura 8.35. El 1 o 0 al lado de cada flecha indica el estado de la entrada de control $UP/\overline{DOWN}$, Y.

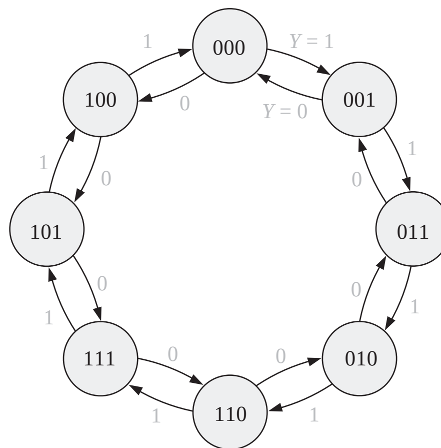


FIGURA 8.35 Diagrama de estados de un contador en código Gray ascendente/ descendente de 3bits.

Paso 2. La tabla del estado siguiente se obtiene a partir del diagrama de estados y se presenta en la Tabla 8.11. Observe que para cada estado actual hay dos posibles estados siguientes, dependiendo de la variable $UP/\overline{DOWN}$ de control, Y.

Paso 3. La tabla de transiciones para el flip-flop J-K se repite en la Tabla 8.12.

Paso 4. Los mapas de Karnaugh para las entradas J y K de los flip-flops se presentan en la Figura 8.36. La entrada de control $UP/\overline{DOWN}$, Y, se considera una de las variables de estado junto con Q_0, Q_1 y Q_2. Utilizando la tabla del estado

siguiente, la información de la columna "Entradas del flip-flop" de la Tabla 8.12 se transfiere a las tablas indicadas para cada estado actual del contador.

Estado actual			Estado siguiente $Y = 0$ (DOWN)			Estado siguiente $Y = 1$ (UP)		
Q_2	Q_1	Q_0	Q_2	Q_1	Q_0	Q_2	Q_1	Q_0
0	0	0	1	0	0	0	0	1
0	0	1	0	0	0	0	1	1
0	1	1	0	0	1	0	1	0
0	1	0	0	1	1	1	1	0
1	1	0	0	1	0	1	1	1
1	1	1	1	1	0	1	0	1
1	0	1	1	1	1	1	0	0
1	0	0	1	0	1	0	0	0

Y = entrada de control UP/$\overline{\text{DOWN}}$.

TABLA 8.11 Tabla del estado siguiente del contador en código Gray ascendente/ descendente de 3 bits.

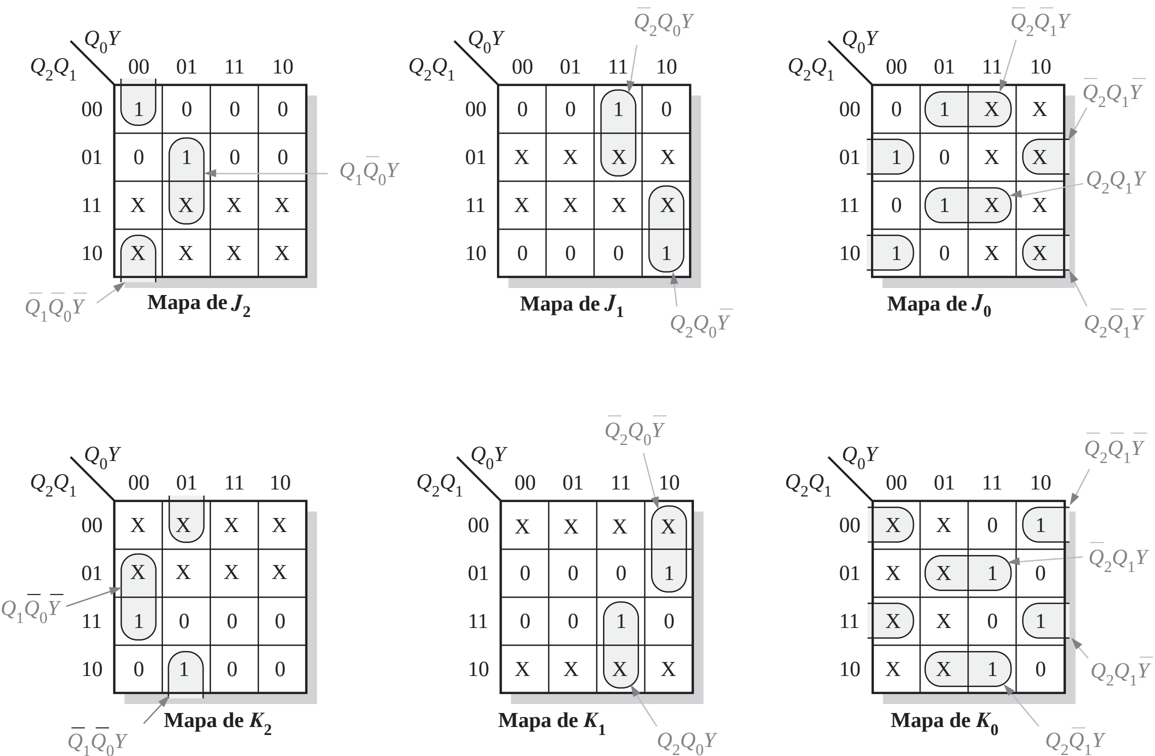


FIGURA 8.36 Mapas para J y K de acuerdo con la Tabla 8.11. La entrada de control UP/$\overline{\text{DOWN}}$, Y, se trata como una cuarta variable.

Transiciones de salida Q_N	Q_{N+1}	Entradas del flip-flop J	K
0 ⟶	0	0	X
0 ⟶	1	1	X
1 ⟶	0	X	1
1 ⟶	1	X	0

TABLA 8.12 Tabla de transiciones del flip-flop J-K.

Paso 5. Los 1s se combinan formando grupos tan grandes como sea posible, utilizando las condiciones indiferentes (X) siempre que se pueda. Se forman los términos producto y se obtienen las expresiones siguientes para las entradas J y K:

$$J_0 = Q_2Q_1Y + Q_2\overline{Q}_1\overline{Y} + \overline{Q}_2\overline{Q}_1Y + \overline{Q}_2Q_1\overline{Y}$$
$$J_1 = \overline{Q}_2Q_0Y + Q_2Q_0\overline{Y}$$
$$J_2 = Q_1\overline{Q}_0Y + \overline{Q}_1\overline{Q}_0\overline{Y}$$

$$K_0 = \overline{Q}_2\overline{Q}_1\overline{Y} + \overline{Q}_2Q_1Y + Q_2\overline{Q}_1Y + Q_2Q_1\overline{Y}$$
$$K_1 = \overline{Q}_2Q_0\overline{Y} + Q_2Q_0Y$$
$$K_2 = Q_1\overline{Q}_0\overline{Y} + \overline{Q}_1\overline{Q}_0Y$$

Paso 6. Las ecuaciones de J y K se implementan con lógica combinacional, obteniendo el contador completo que se muestra en la Figura 8.37 en la página siguiente.

Problema relacionado. Verificar que la lógica de la Figura 8.37 concuerda con las expresiones del paso 5.

REVISIÓN DE LA SECCIÓN 8.4

1. Un flip-flop J- K se encuentra actualmente en estado RESET y tiene que pasar al estado SET en el siguiente impulso de reloj. ¿Cuáles tienen que ser los valores de J y K?
2. Un flip-flop J-K se encuentra actualmente en estado SET y tiene que permanecer en dicho estado durante el siguiente impulso de reloj. ¿Cuáles tienen que ser los valores de J y K?
3. Un contador binario se encuentra en el estado $Q_3\overline{Q}_2Q_1\overline{Q}_0 = 1010$.

   (a) ¿Cuál es el estado siguiente?

   (b) ¿Qué condición tiene que existir en cada entrada de los flip-flops para asegurar que pasa al estado siguiente correcto con el impulso de reloj?

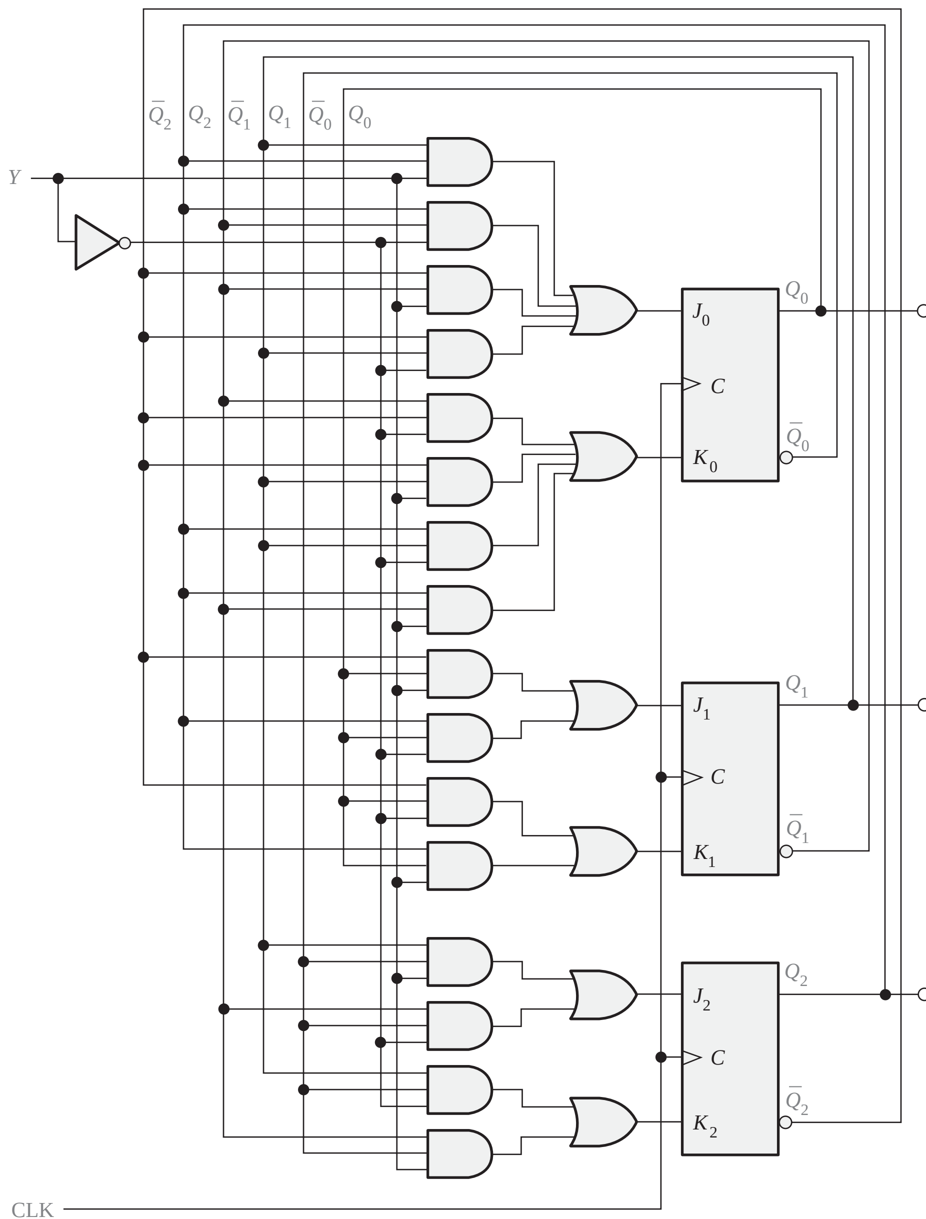


FIGURA 8.37 Contador de código Gray ascendente/descendente de tres bits.

8.5 CONTADORES EN CASCADA

Los contadores se pueden conectar en cascada para conseguir trabajar con módulos mayores. En esencia, conexión en cascada significa que la salida de la última etapa de un contador excita la entrada del siguiente contador.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Determinar el módulo global de los contadores en cascada. ■ Analizar el diagrama de tiempos de una configuración de contadores en cascada. ■ Utilizar contadores en cascada como divisores de frecuencia. ■ Utilizar contadores en cascada para conseguir secuencias específicas truncadas.

▲ *El módulo global de los contadores en cascada es igual al producto de los módulos individuales.*

Un ejemplo de dos contadores conectados en **cascada** se muestra en la Figura 8.38, para el caso de dos contadores con propagación de 2 y 3 bits. El diagrama de tiempos se puede ver en la Figura 8.39. Observe que en el diagrama de tiempos, la salida final del contador de módulo 8, Q_4, se produce una vez por cada 32 impulsos de reloj de entrada. El módulo global de los contadores en cascada es 32, es decir, actúan como un contador de división por 32.

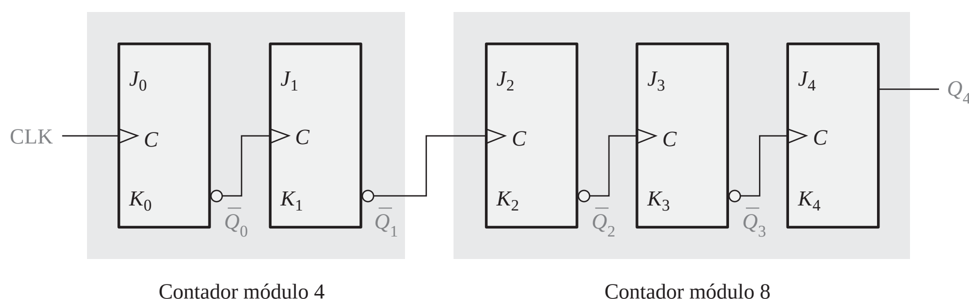


FIGURA 8.38 Dos contadores en cascada (todas las entradas *J* y *K* están a nivel ALTO).

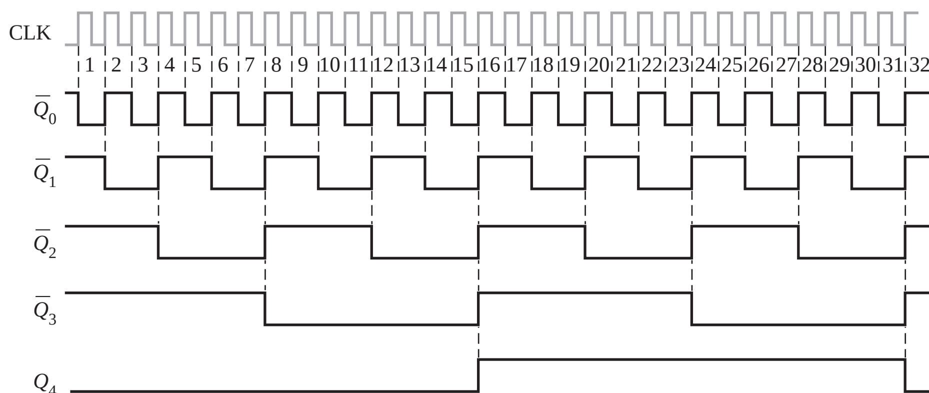


FIGURA 8.39 Diagrama de tiempos para la configuración de contadores en cascada de la Figura 8.38.



NOTAS INFORMÁTICAS

El contador de marca temporal (TSC) mencionado en la anterior nota informática, es un contador de 64 bits. Es interesante observar que si a este contador (o cualquier contador de 64 bits de módulo completo) se le aplica una frecuencia de 100 MHz, tardará 5.849 años en pasar por todos sus estados y alcanzar su valor de cuenta terminal. Por el contrario, un contador de 32 bits de módulo completo recorrerá todos sus estados en aproximadamente 43 segundos cuando se aplique una señal de reloj de 100 MHz. La diferencia es asombrosa.

Cuando se conectan contadores síncronos en una configuración en cascada, es necesario utilizar las funciones de habilitación de cuenta y de fin de cuenta para conseguir trabajar con módulos mayores. En algunos dispositivos, la habilitación de cuenta se denomina simplemente *CTEN* (*Count Enable*) o con cualquier otra designación como *G*, y la señal de fin de cuenta (*TC, Terminal Count)* es análoga a la salida de propagación de reloj (*RCO*) de algunos circuitos integrados contadores.

La Figura 8.40 muestra dos contadores de décadas conectados en cascada. El valor de fin de cuenta (*TC*) del contador 1 se conecta a la entrada de habilitación de cuenta (*CTEN*) del contador 2. El contador 2 se inhibe cuando su entrada *CTEN* está a nivel BAJO, hasta que el contador 1 alcanza su estado final y la salida del valor de fin de cuenta pasa a nivel ALTO. Este nivel ALTO activa ahora el contador 2, de modo que, cuando se produzca el primer impulso de reloj después de que el contador 1 alcance su valor de fin de cuenta (CLK10), el contador 2 pasa de su estado inicial a su segundo estado. Al terminar el segundo ciclo del contador 1 (cuando el contador 1 alcanza el valor de fin de cuenta por segunda vez), el contador 2 se encuentra de nuevo activado y avanza al estado siguiente. Esta secuencia se repite indefinidamente. Dado que se trata de contadores de décadas, el contador 1 tiene que pasar por diez ciclos completos antes de que el contador 2 complete su primer ciclo. En otras palabras, por cada diez ciclos del contador 1, el contador 2 realiza un único ciclo. Por tanto, el contador 2 completará un ciclo después de 100 impulsos de reloj. El módulo global de estos dos contadores en cascada es $10 \times 10 = 100$.

Si lo consideramos como un divisor de frecuencia, el circuito de la Figura 8.40 divide la frecuencia de entrada de reloj entre 100. Los contadores en cascada se utilizan a menudo para dividir una señal de reloj de alta frecuencia, y obtener impulsos de frecuencias precisas. Las configuraciones de los contadores en cascada utilizadas para estos propósitos se denominan algunas veces *cadenas de división*. Por ejemplo, suponga que tenemos una frecuencia de reloj básica de 1 MHz y que se desea obtener 100 kHz, 10 kHz y 1 kHz. Para ello, se pueden utilizar una serie de contadores de décadas en cascada. Si la señal de 1 MHz se divide entre 10, la salida tendrá una frecuencia de 100 kHz. Si dividimos después la señal de 100 kHz entre 10, la salida será una señal de 10 kHz. Otra división por 10 dará la señal de 1 kHz. La implementación de estas cadenas de división se muestra en la Figura 8.41.

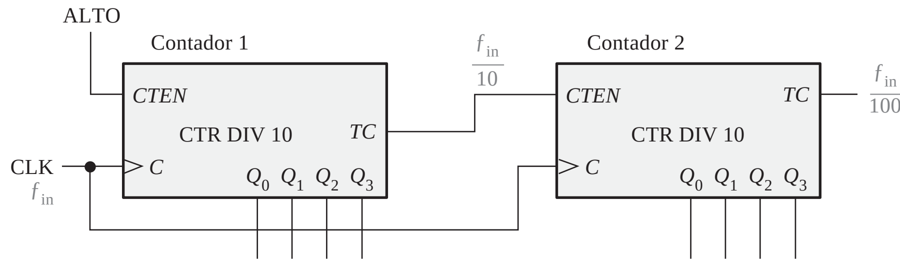


FIGURA 8.40 Contador de módulo 100, que utiliza dos contadores de décadas en cascada.

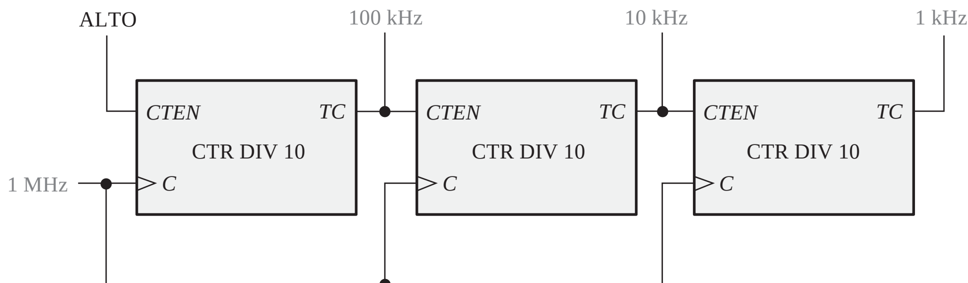


FIGURA 8.41 Tres contadores de décadas en cascada forman un divisor de frecuencia por 1000, con salidas intermedias de división por 100 y por 10.

EJEMPLO 8.7

Determinar el módulo global de las dos configuraciones de contadores en cascada de la Figura 8.42.

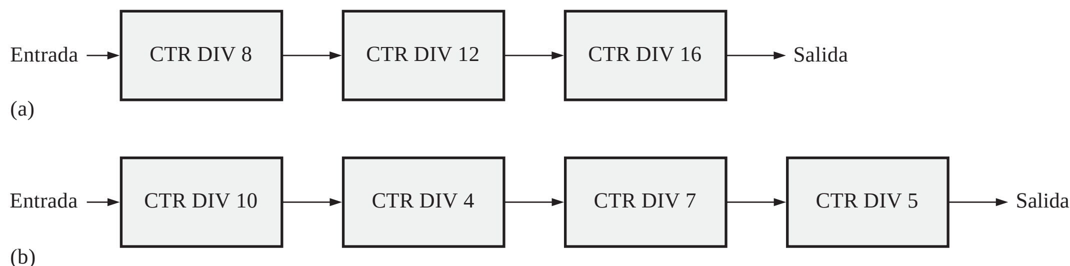


FIGURA 8.42

Solución En la Figura 8.42(a), el módulo global para la configuración de tres contadores es:

$$8 \times 12 \times 16 = \mathbf{1536}$$

En la Figura 8.42(b), el módulo global para la configuración de cuatro contadores es:

$$10 \times 4 \times 7 \times 5 = \mathbf{1400}$$

Problema relacionado ¿Cuántos contadores de décadas en cascada son necesarios para dividir una frecuencia de reloj entre 100.000?

EJEMPLO 8.8

Utilizar contadores 74F162 para obtener una señal de 10 kHz a partir de una señal de reloj de 1 MHz. Dibujar el diagrama lógico.

Solución

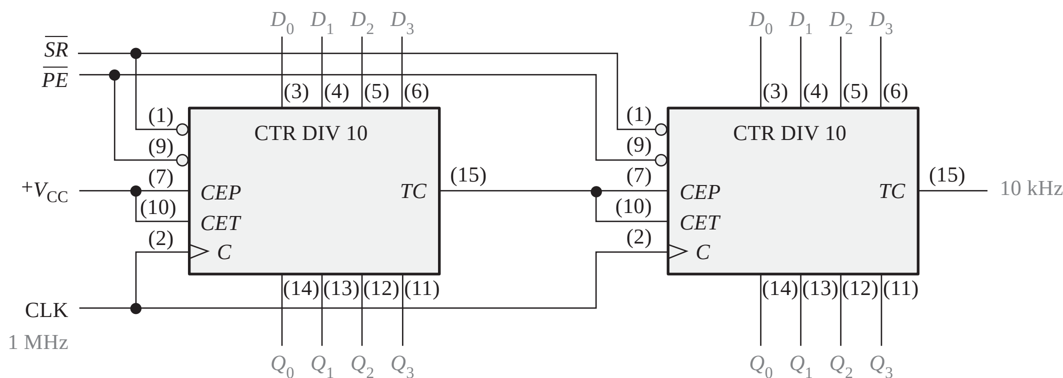


FIGURA 8.43 Contador divisor por 100 que utiliza dos contadores de décadas 74F162.

Para obtener 10 kHz a partir de una señal de reloj de 1 MHz se requiere un factor de división de 100. Se tienen que conectar en cascada dos contadores 74F162, como se muestra en la Figura 8.43. El contador de la izquierda produce un impulso *TC* por cada 10 impulsos de reloj. El contador de la derecha produce un impulso *TC* por cada 100 impulsos de reloj.

Problema relacionado Determinar la frecuencia de la onda de la salida Q_0 del segundo contador (el de la derecha) en la Figura 8.43.

Contadores en cascada con secuencias truncadas

El estudio precedente ha mostrado cómo conseguir un módulo global (factor de división) que sea igual al producto de los módulos individuales de los contadores conectados en cascada. Esto se denomina *conexión en cascada de módulo completo*.

A menudo, una aplicación requiere un módulo global menor que el que se puede conseguir con la conexión en cascada de módulo completo. Es decir, se tiene que implementar una secuencia truncada con contadores en cascada. Para ilustrar este método, utilizaremos la configuración de contadores en cascada de la Figura 8.44. Este circuito particular utiliza cuatro contadores binarios síncronos de 4 bits 74HC161. Si estos cuatro contadores (dieciséis bits en total) se dispusieran en una conexión en cascada de módulo completo, el módulo sería:

$2^{16} = 65.536$

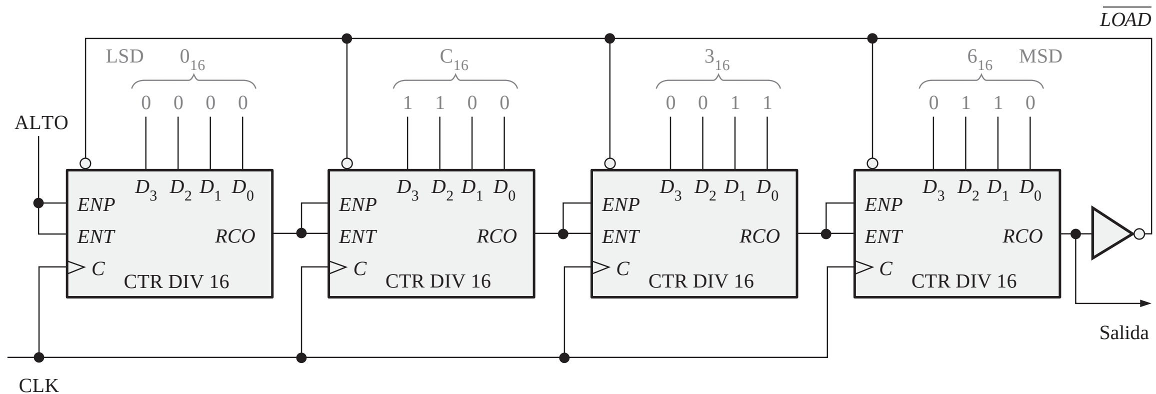


FIGURA 8.44 Contador divisor por 40.000 que utiliza contadores binarios de 4 bits 74HC161. Observe que las entradas paralelas de datos se muestran en orden binario (el bit más a la derecha, D_0, es el LSB de cada contador).

Vamos a suponer que una cierta aplicación requiere un contador-divisor por 40.000 (módulo 40.000). La diferencia entre 65.536 y 40.000 es 25.536, que es el número de estados que tienen que ser *eliminados* de la secuencia de módulo completo. La técnica utilizada en el circuito de la Figura 8.44 sirve para inicializar los contadores en cascada en el estado 25.536 (63C0 en hexadecimal) cada vez que se inicia un nuevo ciclo, de forma que cuenten desde 25.536 hasta 65.536 en cada ciclo completo. Por tanto, cada ciclo completo del contador tiene 40.000 estados.

Observe en la Figura 8.44 que la salida *RCO* del contador más a la derecha está invertida y se aplica a la entrada $\overline{LOAD}$ de cada contador de 4 bits. Cada vez que el contador alcanza su valor de fin de cuenta de

65.535, que es 1111111111111111_2, *RCO* se pone a nivel ALTO y origina que el número que hay en sus entradas paralelas de datos ($63C0_{16}$) se cargue en el contador síncrono con el impulso de reloj. Por tanto, se produce un impulso *RCO* en el contador de 4 bits de más a la derecha por cada 40.000 impulsos de reloj.

Con esta técnica, se puede conseguir cualquier módulo, cargando el contador con el estado inicial apropiado en cada ciclo.

REVISIÓN DE LA SECCIÓN 8.5

1. ¿Cuántos contadores de décadas son necesarios para implementar un contador divisor por 1000 (módulo 1000)? ¿Y uno divisor por 10.000?
2. Mostrar mediante diagramas de bloques generales cómo conseguir cada uno de los siguientes dispositivos, utilizando un flip-flop, un contador de décadas, un contador binario de 4 bits o cualquier combinación de éstos:
   (a) Contador divisor por 20
   (b) Contador divisor por 32
   (c) Contador divisor por 160
   (d) Contador divisor por 320

8.6 DECODIFICACIÓN DE CONTADORES

En muchas aplicaciones, es necesario decodificar algunos o todos los estados del contador. La decodificación de un contador implica la utilización de decodificadores o de puertas lógicas para determinar cuándo se encuentra el contador en un determinado estado binario de su secuencia. Por ejemplo, la función de fin de cuenta estudiada previamente es una decodificación de un único estado (el último estado) de la secuencia del contador.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Implementar la lógica de decodificación para cualquier estado de la secuencia de un contador. ■ Explicar por qué aparecen *glitches* en la lógica de decodificación de un contador. ■ Utilizar el método de validación (*strobing*) para eliminar los *glitches* en la decodificación.

Supongamos que se desea decodificar el estado binario 6 (110) de un contador binario de 3 bits. Cuando $Q_2 = 1$, $Q_1 = 1$ y $Q_0 = 0$, aparece un nivel ALTO en la salida de la puerta de decodificación, indicando que el

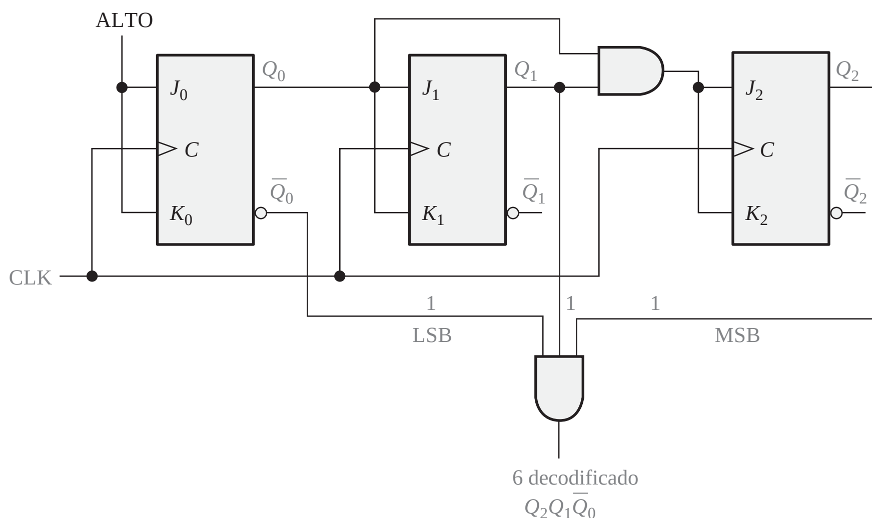


FIGURA 8.45 Decodificación del estado 6 (110).

EJEMPLO 8.9

Implementar la decodificación de los estados binarios 2 y 7 de un contador síncrono de 3 bits. Dibujar el diagrama de tiempos completo y las formas de onda de salida de las puertas de decodificación. 2 binario = $\overline{Q}_2 Q_1 \overline{Q}_0$ y 7 binario = $Q_2 Q_1 Q_0$.

Solución *Véase* la Figura 8.46. El contador de 3 bits fue explicado anteriormente en la Sección 8.2 (Figura 8.14).

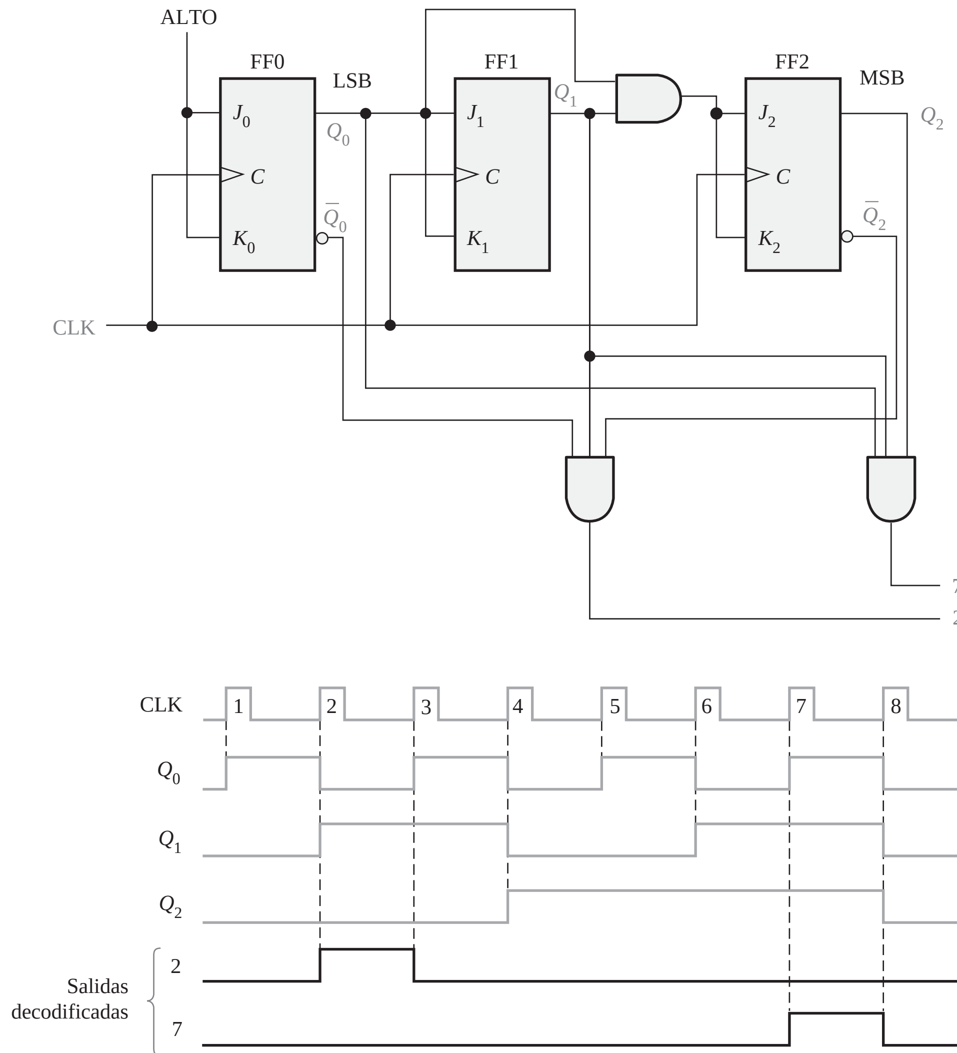


FIGURA 8.46 Contador de 3 bits con decodificación activa a nivel ALTO de los estados 2 y 7.

Problema relacionado Realizar la lógica de decodificación del estado 5 de un contador de 3 bits.

contador se encuentra en el estado 6. Esto se puede realizar como se muestra en la Figura 8.45. Esto se denomina *decodificación activa a nivel ALTO.* Reemplazando la puerta AND por una puerta NAND obtenemos una decodificación activa a nivel BAJO.

Glitches en la decodificación

▲ *Un glitch es un pico de tensión no deseado.*

En el Capítulo 6 se introdujo el problema de los *glitches* producidos por el proceso de decodificación. Como ya se ha visto, los retardos de propagación debidos al efecto del retraso en los contadores asíncronos origina estados transitorios, en los que las salidas del contador están variando en instantes de tiempo ligeramente distintos. Estos estados transitorios producen picos de tensión de corta duración (*glitches*) no deseados, que aparecen en las salidas del decodificador conectado al contador. El problema de los *glitches* puede también aparecer en cierta medida en los contadores síncronos, ya que los retardos de propagación entre el reloj y las salidas *Q* de cada flip-flop del contador pueden diferir ligeramente.

La Figura 8.47 muestra un contador asíncrono básico de décadas BCD conectado a un decodificador BCD-decimal. Para ver qué es lo que ocurre, vamos a examinar el diagrama de tiempos de la Figura 8.48, en el que se tienen en cuenta los retardos de propagación. Observe que estos retardos originan estados erróneos de corta duración. El valor del estado binario falso en cada transición crítica se indica en el diagrama. Los *glitches* resultantes pueden verse en las salidas del decodificador.

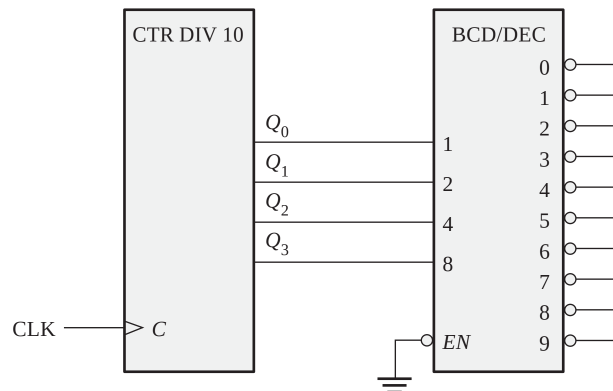


FIGURA 8.47 Contador básico de décadas (BCD) y decodificador.

Una manera de eliminar los *glitches* consiste en activar las salidas del decodificador después de que los impulsos de ruido hayan tenido tiempo de desaparecer. Este método se conoce como *validación (strobing)* y en el caso de una señal de reloj activa a nivel ALTO se puede implementar como se muestra en la Figura 8.49, utilizando el nivel BAJO del reloj para activar el decodificador. El diagrama de tiempos mejorado que se obtiene se presenta en la Figura 8.50.

REVISIÓN DE LA SECCIÓN 8.6

1. ¿Cuáles son los posibles estados transitorios cuando un contador binario de 4 bits asíncrono cambia del

(a) estado 2 al 3?

(b) estado 3 al 4?

(c) estado 10_{10} al 11_{10}?

(d) estado 15 al 0?

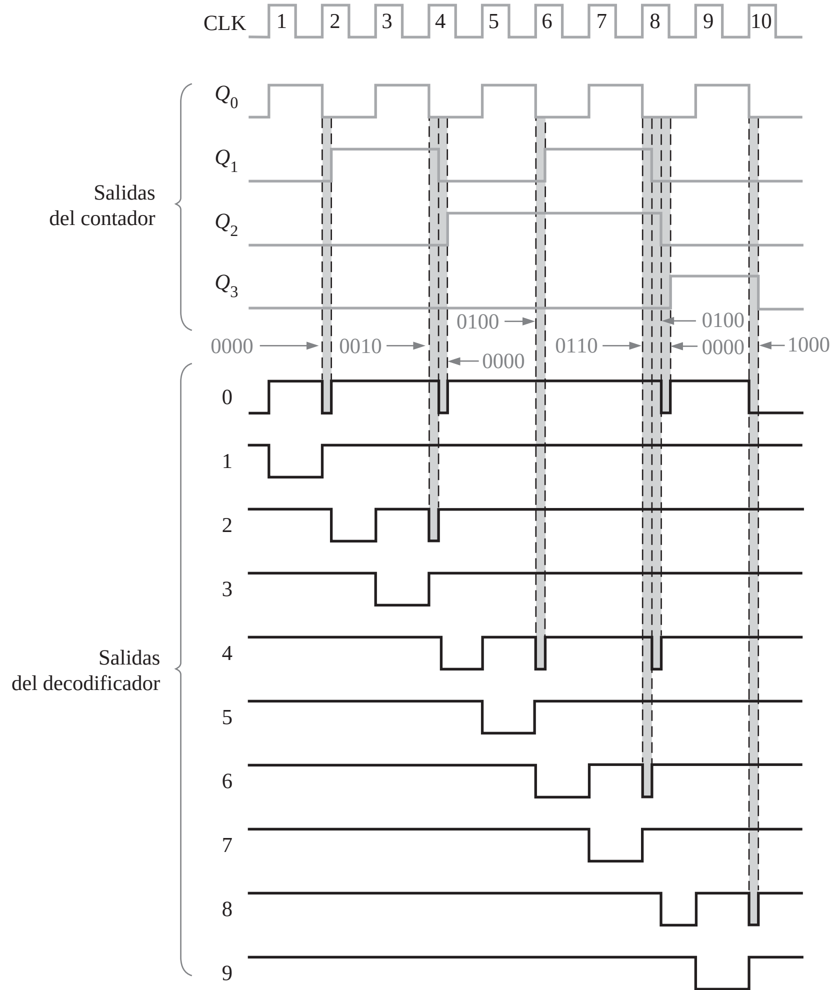


FIGURA 8.48 Salidas con *glitches* procedentes del decodificador de la Figura 8.47. Las anchuras de los *glitches* están exageradas y, generalmente, tienen un valor de unos pocos nanosegundos.

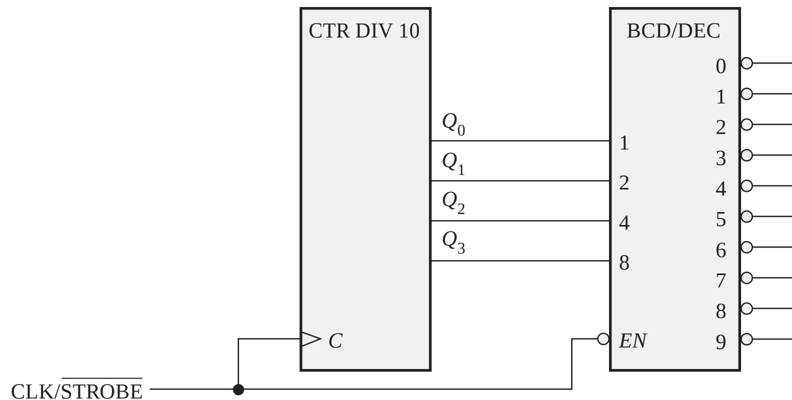


FIGURA 8.49 Contador básico de décadas y decodificador con validación (*strobe*) para eliminar los *glitches*.

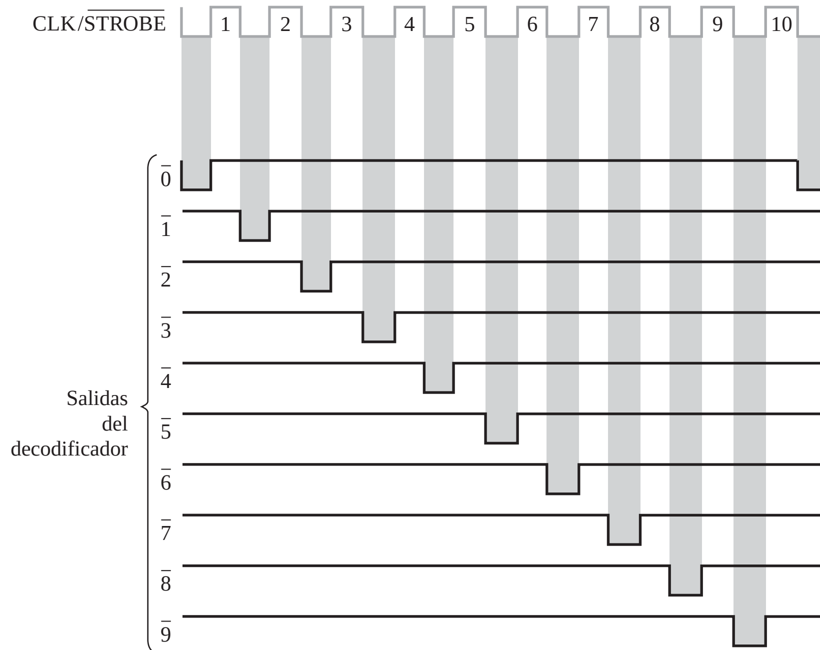


FIGURA 8.50 Salidas del decodificador con validación para el circuito de la Figura 8.49.

8.7 APLICACIONES DE LOS CONTADORES

Los contadores digitales son dispositivos muy versátiles y útiles que se pueden encontrar en muchas aplicaciones. En esta sección, se presentan varias aplicaciones representativas de los contadores.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Describir cómo se utilizan los contadores en un sistema básico de reloj digital. ■ Explicar cómo se implementa un contador-divisor por 60 y cómo se utiliza en un reloj digital. ■ Explicar cómo se implementan los contadores de horas. ■ Estudiar cómo aplicar un contador en un sistema de control de un aparcamiento de automóviles. ■ Describir cómo se utiliza un contador en un proceso de conversión de datos paralelo-serie

El reloj digital

Un ejemplo típico de aplicación de los contadores son los sistemas de control de tiempo. La Figura 8.51 es un diagrama lógico simplificado de un reloj digital, que presenta en el display segundos, minutos y horas. En primer lugar, se transforma una tensión alterna sinusoidal de 60 Hz en un tren de impulsos a 60 Hz y, posteriormente, se divide para obtener un tren de impulsos a 1 Hz, mediante un contador-divisor por 60, formado por un divisor por 10 seguido de un un divisor por 6. La cuenta de *minutos* y de *segundos* se genera también mediante contadores-divisores por 60, operación que se puede ver en detalle en la Figura 8.52. Estos contadores cuentan desde 0 hasta 59 y luego vuelven al estado 0; en esta implementación particular se utilizan contadores de décadas síncronos. Observe que la etapa del divisor por 6 está constituida por un contador de décadas con una secuencia truncada, que se logra utilizando el estado 6 decodificado para borrar, en modo asíncrono, el contador. El valor de final de cuenta 59 también se codifica para activar el siguiente contador de la cadena.

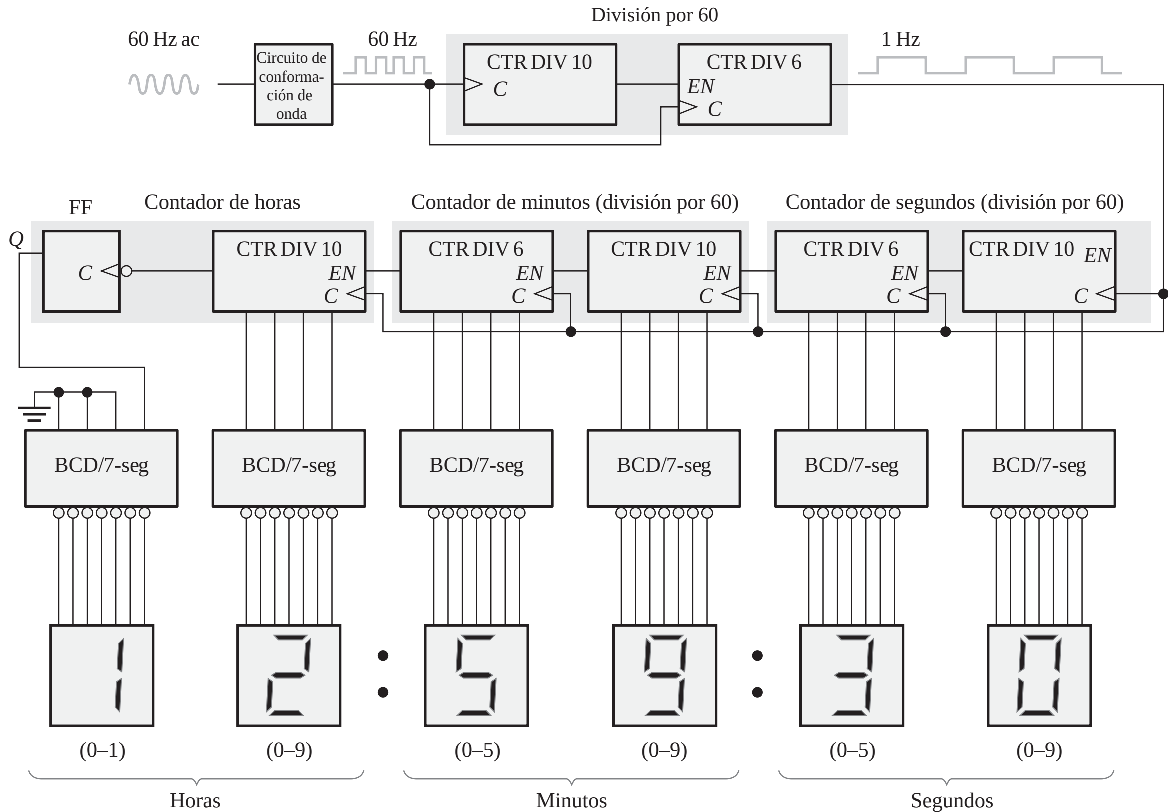


FIGURA 8.51 Diagrama lógico simplificado de un reloj digital de 12 horas. Los detalles del circuito lógico, con sus dispositivos específicos, se muestran en las Figuras 8.52 y 8.53.

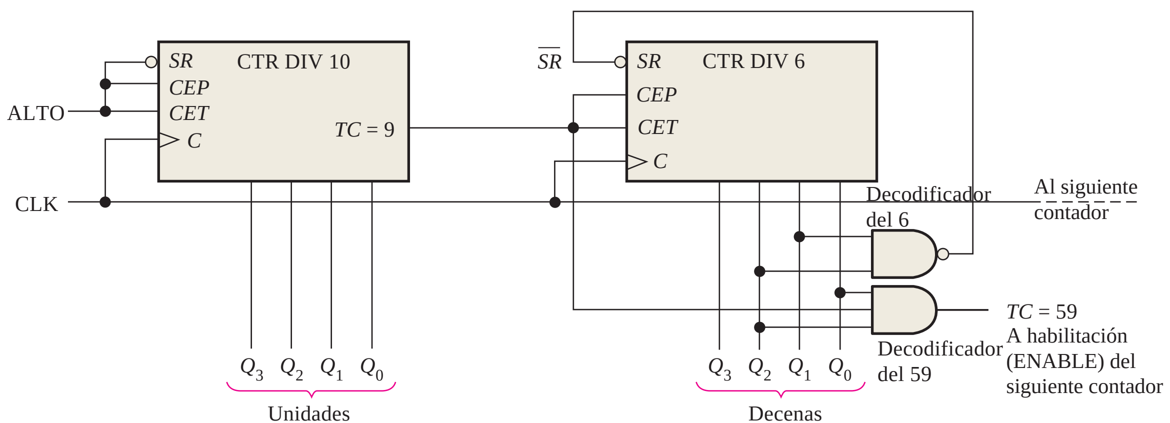


FIGURA 8.52 Diagrama lógico de un contador-divisor por 60 que utiliza contadores de décadas síncronos 74F162. Observe que las salidas están en orden binario (el bit más a la derecha es el menos significativo, LSB).

El contador de *horas* se implementa mediante un contador de décadas y un flip-flop, tal y como se muestra en la Figura 8.53. Suponga que, inicialmente, tanto el contador de décadas como el flip-flop se encuentran

en estado RESET, y que la puerta de decodificación del estado 12 está en nivel ALTO. El contador de décadas avanza pasando por todos sus estados desde cero hasta nueve y, al pasar de nueve a cero para iniciar un nuevo ciclo, el flip-flop bascula al estado SET ($J = 1$, $K = 0$). Esto hace que se ilumine un 1 en el display, que indica el dígito de las decenas de horas. El valor total de cuenta es ahora 10 (el contador de décadas está en estado cero y el flip-flop en estado SET).

A continuación, el número total avanza hasta once y luego a doce. En el estado 12, la salida Q_2 del contador de décadas es un nivel ALTO, el flip-flop sigue en estado SET y, por tanto, la salida de la puerta 12 de decodificación está a nivel BAJO. Esto activa la entrada $\overline{PE}$ del contador de décadas. En el siguiente impulso de reloj, el contador de décadas es inicializado en el estado 1 a través de las entradas de datos, y el flip-flop pasa al estado de RESET ($J = 0$, $K = 1$). Como puede ver, esta lógica hace que siempre el contador inicie un nuevo ciclo pasando de doce a uno, en lugar de a cero.

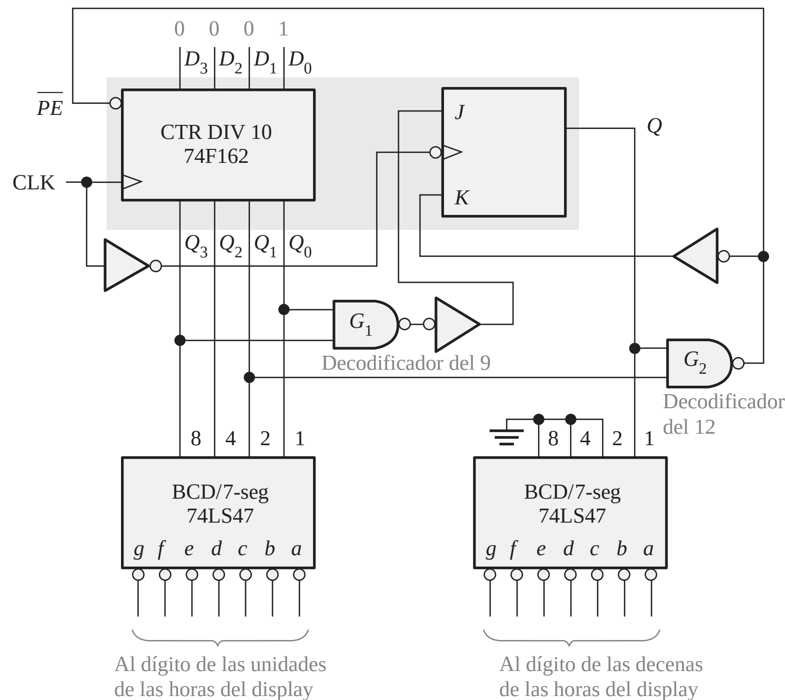


FIGURA 8.53 Diagrama lógico del contador y de los decodificadores de horas. Observe que en las entradas y salidas del contador, el bit más a la derecha es el LSB.

Sistema de control de un aparcamiento de automóviles

Ahora vamos a ver una sencilla aplicación que ilustra cómo puede resolver un contador ascendente/descendente un problema cotidiano. El problema consiste en concebir una forma de control de las plazas disponibles en un aparcamiento de 100 plazas y, en caso de que esté lleno, hacer que se encienda una luz de aviso y que se baje una barrera a la entrada.

Un sistema que resuelve este problema está constituido por: (1) sensores optoelectrónicos en la entrada y salida del aparcamiento, (2) un contador ascendente/descendente y su circuitería asociada y, (3) un circuito de interfaz que utilice la salida del contador para encender o apagar la luz de COMPLETO, así como para subir o bajar la barrera de entrada. En la Figura 8.54 se presenta un diagrama general de bloques de este sistema.

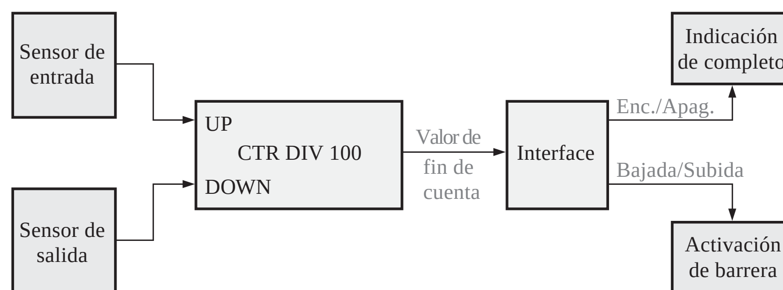


FIGURA 8.54 Diagrama de bloques funcional para el control del aparcamiento.

En la Figura 8.55 se muestra un diagrama lógico del contador ascendente/descendente. Está formado por dos contadores de décadas ascendentes/descendentes 74HC190 conectados en cascada. Su funcionamiento se describe en los siguientes párrafos.

▲ *Incrementar un contador es sumar uno al valor de cuenta.*

El contador se inicializa previamente en el estado 0 mediante las entradas de datos en paralelo, las cuales no se muestran. Cada automóvil que entra en el aparcamiento interrumpe la trayectoria de un haz de luz, activando un sensor que produce un impulso eléctrico. Este impulso positivo activa el *latch* S-R con su flanco anterior. El nivel BAJO en la salida $\overline{Q}$ del *latch* hace que el contador entre en el modo de trabajo ascendente. Además, el impulso producido por el sensor pasa a través de la puerta NOR y aplica la señal de reloj al contador durante la transición de nivel BAJO a nivel ALTO de su flanco posterior. Cada vez que entra un coche en el aparcamiento, el contador avanza una posición (**se incrementa**). Cuando han entrado cien automóviles, el contador llega a su estado final (100_{10}). La salida *MAX/MIN* se pone a nivel ALTO y activa el circuito de interfaz (el cual no se detalla), que enciende la luz de COMPLETO y baja la barrera para evitar que sigan entrando coches.

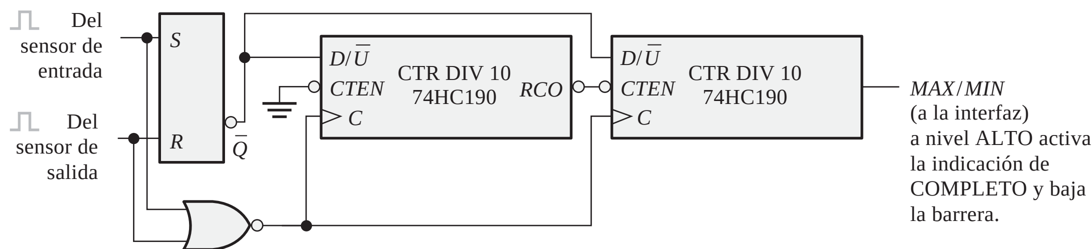


FIGURA 8.55 Diagrama lógico de un contador ascendente/descendente de módulo 100 para el control del aparcamiento.

▲ *Decrementar un contador es restar uno al valor de cuenta.*

Cuando sale un automóvil, un sensor optoelectrónico produce un impulso positivo, que pone a cero (RESET) el *latch* S-R y hace que el contador entre en modo descendente. El flanco posterior del reloj **decrementa** en una unidad el número que marque el contador. Si el aparcamiento está completo y sale un automóvil, la salida *MAX/MIN* del contador pasa a nivel BAJO, haciendo que desaparezca la luz de COMPLETO y subiendo la barrera.

Conversión de datos paralelo-serie (multiplexación)

En el Capítulo 6 ya expusimos un ejemplo simplificado de transmisión de datos utilizando técnicas de multiplexación y demultiplexación. Esencialmente, los bits de datos paralelos en las entradas del multiplexor se

convierten en bits de datos serie que se transmiten por una única línea. Se denominan *datos en paralelo* a un grupo de bits que se presentan simultáneamente sobre varias líneas paralelas. Se denominan *datos en serie* a un grupo de bits que se presentan secuencialmente por una única línea.

Normalmente, la conversión paralelo-serie se realiza utilizando un contador que proporcione una secuencia binaria para las entradas de selección de datos de un multiplexor/selector de datos, como muestra la Figura 8.56. Las salidas Q del contador de módulo 8 se conectan a las entradas de selección de datos de un multiplexor de 8 bits.

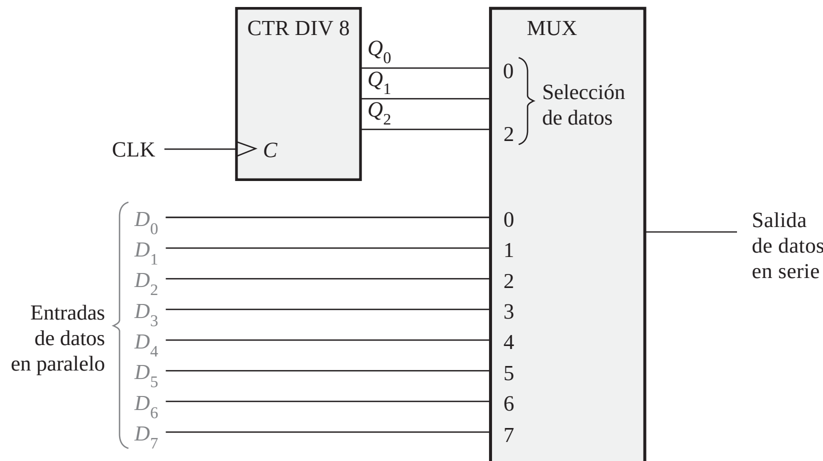
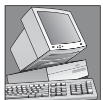


FIGURA 8.56 Lógica de conversión de datos paralelo-serie.

La Figura 8.57 es un diagrama de tiempos que muestra el funcionamiento de este circuito. El primer byte (grupo de ocho bits) de datos en paralelo se aplica a las entradas del multiplexor. A medida que el contador avanza a lo largo de su secuencia binaria desde cero hasta siete, cada bit, comenzando por D_0, se selecciona secuencialmente y pasa a través del multiplexor hasta la línea de salida. Después de ocho impulsos de reloj, el byte de datos ha sido convertido a formato serie y enviado a través de la línea de transmisión. Cuando el contador inicia un nuevo ciclo retornando a 0, el siguiente byte se aplica a las entradas de datos y se convierte secuencialmente en formato serie a medida que el contador pasa por sus ocho estados. Este proceso continúa repetidamente para convertir cada byte paralelo a serie.



NOTAS INFORMÁTICAS

Las computadoras disponen de un contador interno que puede programarse para distintas frecuencias y duraciones de tonos, produciendo "música". Para seleccionar un tono concreto, la instrucción programada selecciona un valor divisor que es enviado al contador. El divisor configura al contador de modo que divida la frecuencia básica del reloj del periférico, para generar un tono de audio. La duración de un tono también se puede definir mediante una instrucción de programa; por tanto, se utiliza un contador básico para generar melodías controlando la frecuencia y duración de los tonos.

REVISIÓN DE LA SECCIÓN 8.7

1. Explicar para qué sirve cada puerta NAND de la Figura 8.53.
2. Identificar las dos condiciones para iniciar un nuevo ciclo en el contador de horas de la Figura 8.51, y explicar para qué sirven.

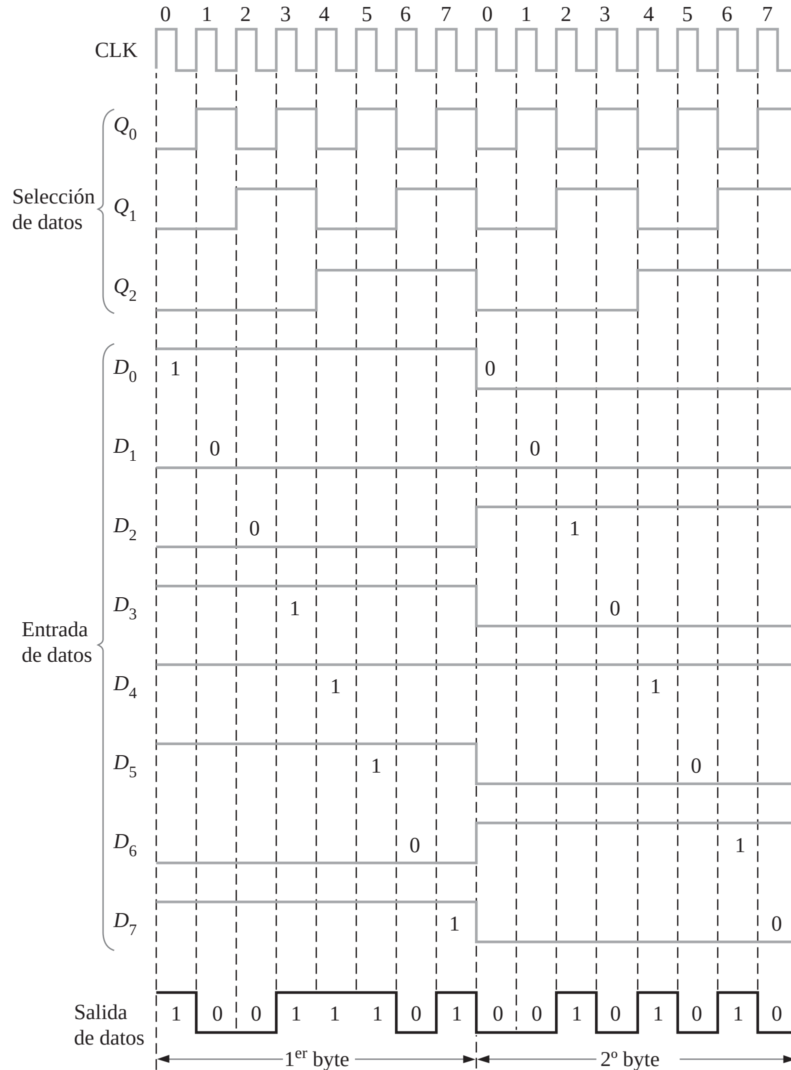


FIGURA 8.57 Diagrama de tiempos de la conversión paralelo-serie del circuito de la Figura 8.56.

8.8 SÍMBOLOS LÓGICOS CON NOTACIÓN DE DEPENDENCIA

Hasta este momento, se han introducido de una manera limitada los símbolos lógicos con notación de dependencia especificados en las normas ANSI/IEEE Standard 91-1984. En muchos casos, los nuevos símbolos no se apartan significativamente de los tradicionales. Sin embargo, existen diferencias considerables en algunos dispositivos, incluidos los contadores y otros dispositivos más complejos, con respecto a lo que estamos acostumbrados a ver. Aunque vamos a continuar utilizando principalmente los símbolos más tradicionales y familiares a lo largo del libro, se proporciona una breve descripción de los símbolos lógicos con notación de dependencia. Se utiliza como ejemplo un circuito integrado contador específico.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Interpretar los símbolos lógicos que incluyen notación de dependencia. ■ Interpretar el bloque común y los elementos individuales del símbolo de un contador. ■ Interpretar el símbolo de cualificación. ■ Explicar la dependencia de control. ■ Explicar la dependencia de modo. ■ Explicar la dependencia AND.

La notación de dependencia es fundamental en el estándar ANSI/IEEE. La notación de dependencia se utiliza junto con los símbolos lógicos para especificar las relaciones entre entradas y salidas, de forma que el funcionamiento lógico de un dispositivo específico pueda ser determinado enteramente a partir de su símbolo lógico, sin ningún conocimiento a priori de los detalles de su estructura interna y sin necesidad de ningún diagrama lógico detallado como referencia. Esta explicación de un símbolo lógico específico con notación de dependencia tiene el fin de ayudar en la interpretación de otros símbolos de este tipo con los que se puede encontrar en el futuro.

Se utiliza el contador binario síncrono de 4 bits 74HC163 como ejemplo. Para poder comparar, la Figura 8.58 muestra el símbolo de bloque tradicional y el símbolo ANSI/IEEE con notación de dependencia. La descripción básica del símbolo y de la notación de dependencia es la siguiente:

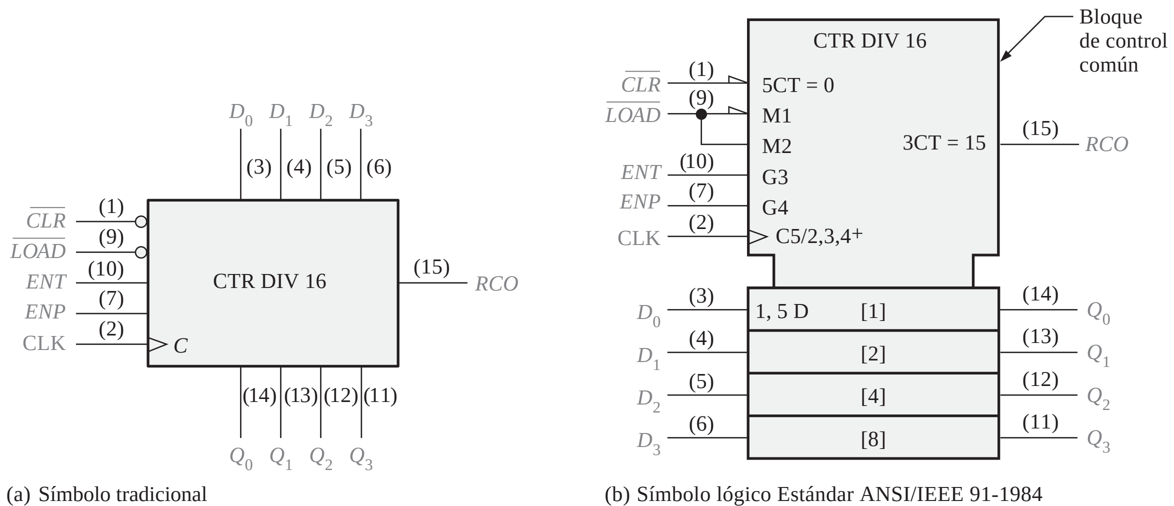


FIGURA 8.58 El contador síncrono de 4 bits 74HC163.

Bloque de control común El bloque superior con las esquinas cortadas de la Figura 8.58(b) tiene varias entradas y una salida que se consideran comunes a todos los elementos del dispositivo y no son exclusivas de ningún elemento.

Elementos individuales El bloque inferior de la Figura 8.58(b), que está dividido en cuatro secciones colindantes, representa los cuatro elementos de almacenamiento (flip-flops D) del contador, cuyas entradas son D_0, D_1, D_2 y D_3, y cuyas salidas son Q_0, Q_1, Q_2,y Q_3.

Símbolo de cualificación La etiqueta "CTR DIV 16" de la Figura 8.58(b) identifica el dispositivo como un contador (CTR) con dieciséis estados (DIV 16).

Dependencia de control (C) Como se muestra en la Figura 8.58(b), la letra *C* denota dependencia del control. Las entradas de control normalmente activan o desactivan las entradas de datos (*D, J, K, S* y *R*) de un elemen-

to de almacenamiento. La entrada C suele ser la entrada de reloj. En este caso, el dígito 5 que sigue a C (C5/2,3,4+) indica que las entradas etiquetadas con un prefijo 5 dependen del reloj (están sincronizadas con el reloj). Por ejemplo, 5CT = 0 en la entrada $\overline{CLR}$ indica que la función de borrado depende del reloj; es decir, es una entrada de borrado síncrona. Cuando la entrada $\overline{CLR}$ está a nivel BAJO (0), el contador se pone a cero (CT = 0) en el flanco de disparo del impulso de reloj. Además, la etiqueta 5D en la entrada del elemento de almacenamiento [1] indica que el almacenamiento de datos depende del reloj (está sincronizado con el reloj). Todas las etiquetas en el elemento de almacenamiento [1] se aplican también a los elementos [2], [4] y [8] que están situados por debajo, ya que no se indica en ellos una notación diferente.

Dependencia de modo (M) Como se muestra en la Figura 8.58(b), la letra M indica dependencia de modo. Se utiliza para indicar cómo dependen las funciones correspondientes a varias entradas o salidas del modo en que está funcionando el dispositivo. En este caso, el dispositivo tiene dos modos de funcionamiento. Cuando la entrada $\overline{LOAD}$ está a nivel BAJO (0), como se indica mediante el triángulo de entrada, el contador está en modo de carga (M1) en el que los datos de entrada (D_0, D_1, D_2 y D_3) se introducen de manera síncrona en los cuatro flip-flops. El dígito 1 detrás de M (M1) y el 1 en la etiqueta 1, 5 D muestran una relación de dependencia y nos indican que los datos de entrada se almacenan sólo cuando el dispositivo está en el modo de carga (M1), en el que $\overline{LOAD} = 0$. Cuando la entrada $\overline{LOAD}$ está a nivel ALTO (1), el contador avanza a través de su secuencia binaria normal, como se indica mediante M2 y el 2 en C5/2,3,4+.

Dependencia AND (G) Como muestra la Figura 8.58(b), la letra G denota dependencia AND, la cual indica que se efectúa una operación AND entre una entrada, designada con la letra G y seguida por un dígito, por un lado, y cualquier otra entrada o salida que tenga el mismo dígito como prefijo en su etiqueta. En este ejemplo en particular, G3 en la entrada ENT y 3CT = 15 en la salida RCO están relacionadas, como indica el dígito 3, y esta relación es una dependencia AND, como indica la G. Esto nos dice que ENT tiene que ser un nivel ALTO (no hay triángulo en la entrada) *y* que el estado del contador debe ser quince (CT = 15) para que la salida RCO sea un nivel ALTO.

Además, los dígitos 2, 3 y 4 en la etiqueta C5/2,3,4+ indican que el contador avanza a través de sus estados cuando $\overline{LOAD} = 1$, como se indica mediante la etiqueta M2 de dependencia de modo, y cuando ENT = 1 y ENP = 1, como se indica por las etiquetas de dependencia AND G3 y G4. El signo más (+) indica que el contador se incrementa cuando existe esta condición.

Esta descripción de un símbolo lógico con notación de dependencia específico pretende ayudar en la interpretación de otros símbolos similares con los que pueda encontrarse en el futuro.

REVISIÓN DE LA SECCIÓN 8.8

1. En la notación de dependencia, ¿qué significan las letras *C, M* y *G?*
2. ¿Qué letra indica almacenamiento de datos?

8.9 LOCALIZACIÓN DE AVERÍAS

La localización de averías en los contadores puede ser simple o muy complicada, dependiendo del tipo de contador y del tipo de fallo. Esta sección nos va a proporcionar un poco de práctica en la resolución de problemas en los circuitos secuenciales.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Detectar fallos en un contador. ■ Aislar los fallos en los contadores en cascada de módulo máximo. ■ Aislar los fallos en los contadores en cascada con secuencia truncada. ■ Determinar los fallos en los contadores implementados con flip-flops individuales.

Contadores

En un contador que tenga una secuencia que no esté controlada por alguna lógica externa, la única cosa que se puede comprobar (aparte de V_{CC} y tierra) es la posibilidad de tener entradas o salidas en circuito abierto o cortocircuitadas. Un CI contador casi nunca altera su secuencia de estados debido a un fallo interno, por lo que sólo se debe comprobar la actividad de los impulsos en las salidas *Q,* para detectar la existencia de circuitos abiertos o cortocircuitos. La ausencia de actividad de impulsos en una de las salidas *Q* indica que hay un circuito abierto o un cortocircuito interno. La ausencia de actividad de impulsos en todas las salidas *Q* indica que la entrada de reloj está fallando o que la entrada de borrado se mantiene en su estado activo.

Para comprobar la entrada de borrado, se aplica un nivel activo constante a la misma a la vez que la señal de reloj. Si se obtiene un nivel BAJO en cada una de las salidas *Q*, el funcionamiento es correcto.

La capacidad de carga paralelo de un contador se puede comprobar activando la entrada de carga paralelo y probando cada uno de los estados del siguiente modo: se aplican niveles bajos a las entradas de datos en paralelo, impulsos en la entrada de reloj y se comprueba que haya niveles bajos en todas las salidas *Q*. A continuación, se aplican niveles altos a las entradas de datos en paralelo, se introducen impulsos en la entrada de reloj y se comprueba que haya niveles altos en todas las salidas *Q*.

Contadores en cascada con módulo máximo

Un fallo en uno de los contadores de una cadena de contadores en cascada puede afectar a todos los demás contadores que le siguen. Por ejemplo, si la entrada de habilitación de cuenta de un contador está en circuito abierto, actúa como si fuera un nivel ALTO (en TTL) y el contador estará siempre activado. Este tipo de fallo en uno de los contadores hará que ese contador funcione a la máxima velocidad de reloj y también que todos los demás contadores que lo sigan funcionen a velocidades mayores de las normales. Esto se ilustra en la Figura 8.59 para un contador-divisor por 1000 en cascada, donde una entrada de habilitación de cuenta (*CTEN)* en circuito abierto actúa como un nivel ALTO TTL y activa continuamente al segundo contador. Otro de los fallos que pueden afectar a las "etapas secundarias" de los contadores pueden ser entradas de reloj o salidas de valor de fin de cuenta en circuito abierto o cortocircuitadas. En algunas de estas situaciones, se

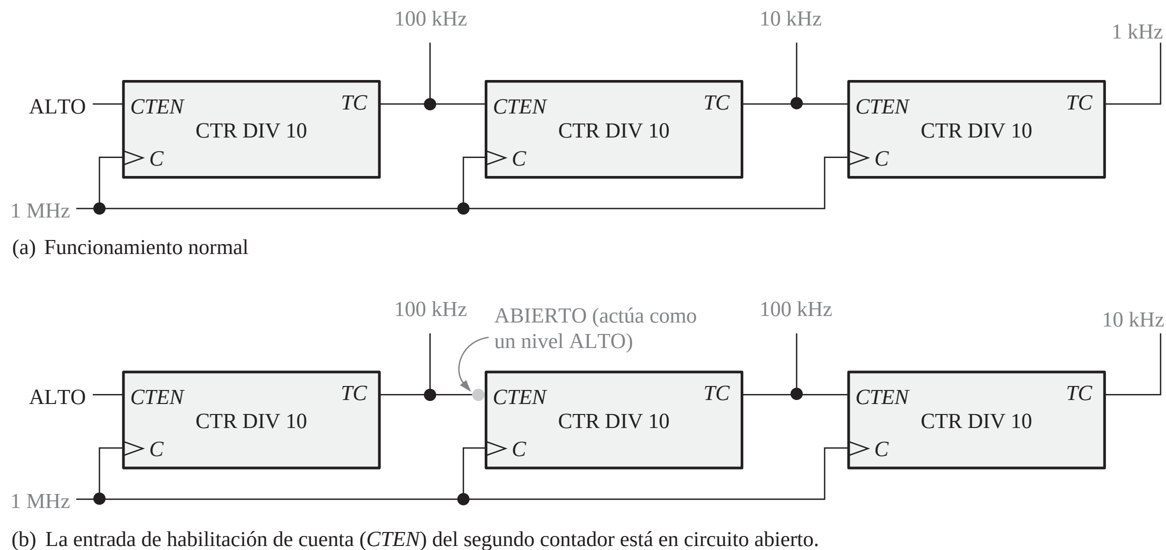


FIGURA 8.59 Ejemplo de fallo que afecta a los sucesivos contadores en una configuración en cascada.

puede observar actividad de impulsos, pero a frecuencias erróneas. En este caso, se tendrán que realizar medidas de frecuencia exactas.

Contadores en cascada con secuencias truncadas

La secuencia de números de un contador en cascada con una secuencia truncada, como el de la Figura 8.60, puede dar lugar a otros tipos de fallos, además de los mencionados para los contadores en cascada de módulo máximo. Por ejemplo, un fallo en una de las entradas de datos en paralelo, la entrada $\overline{LOAD}$ o el inversor pueden alterar el valor de inicialización y, por tanto, cambiar el módulo del contador.

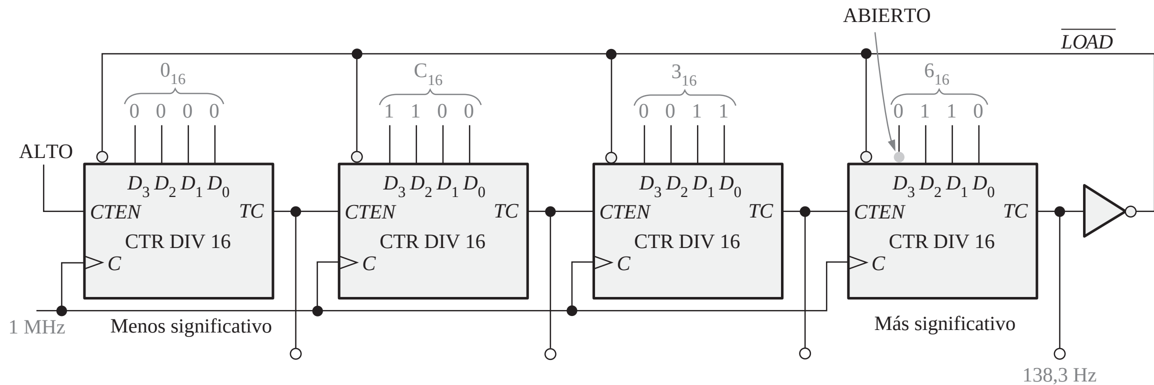


FIGURA 8.60 Ejemplo de fallo en un contador en cascada con secuencia truncada.

Por ejemplo, supongamos que la entrada D_3 del contador más significativo de la Figura 8.60 está en circuito abierto y actúa como un nivel ALTO. Entonces, en lugar de hacer la reinicialización en el estado 6_{16} (0110), se hará en el estado E_{16} (1110). De esta manera, en lugar de comenzar en $63C0_{16}$ (25.536_{10}), cada vez que el contador inicie un nuevo ciclo, la secuencia comenzará en $E3C0_{16}$ (58.304_{10}). Esto hace que varíe el módulo del contador de 40.000 a 65.536 − 58.304 = 7232.

Para comprobar este contador se aplica una frecuencia de reloj conocida, por ejemplo 1 MHz, y se mide la frecuencia de salida en el terminal del valor de fin de cuenta. Si el contador está funcionando adecuadamente, la frecuencia de salida será:

$$f_{out} = \frac{f_{in}}{\text{módulo}} = \frac{1\,\text{MHz}}{40.000} = 25\,\text{Hz}$$

En este caso, el fallo específico descrito en el párrafo anterior hará que la frecuencia de salida sea:

$$f_{out} = \frac{f_{in}}{\text{módulo}} = \frac{1\,\text{MHz}}{7232} = 138{,}3\,\text{Hz}$$

EJEMPLO 8.10

Se realizan medidas de frecuencia en el contador truncado de la Figura 8.61 tal y como se indica. Determinar si el contador está funcionando adecuadamente y, si no es así, determinar cuál es el fallo.

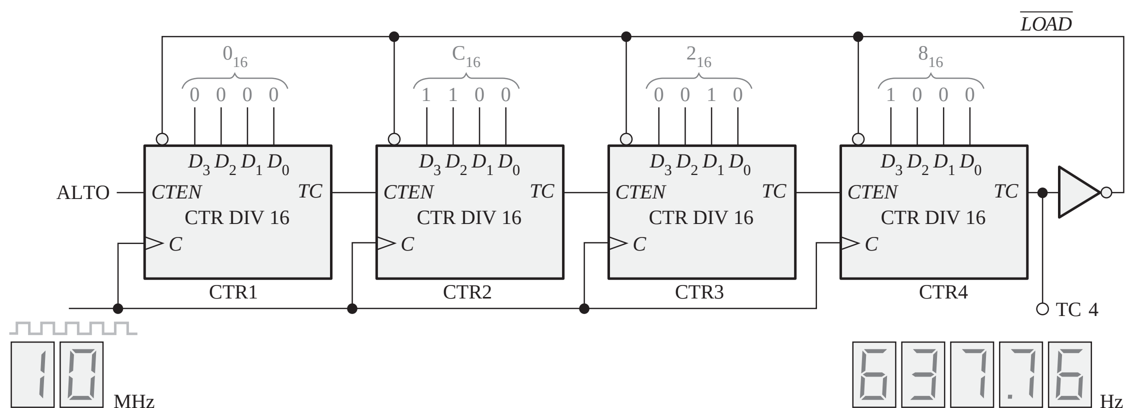


FIGURA 8.61

Solución Se comprueba que la frecuencia medida en TC 4 es la correcta. Si así es, el contador está funcionando perfectamente.

$$
\begin{aligned}
\text{módulo truncado} &= \text{módulo completo} - \text{valor de inicialización} \\
&= 16^4 - 82C0_{16} \\
&= 65.536 - 33.472 = 32.064
\end{aligned}
$$

La frecuencia correcta en TC 4 es

$$f_4 = \frac{10\,\text{MHz}}{32.064} = 311{,}88\text{ Hz}$$

Sin embargo, al medir, detectamos que hay un problema. La frecuencia medida de 637,76 Hz no concuerda con la frecuencia correcta calculada de 311,88 Hz.

Para comprobar el contador que falla, se determina el módulo truncado real del siguiente modo:

$$\text{módulo} = \frac{f_{\text{in}}}{f_{\text{out}}} = \frac{10\,\text{MHz}}{637{,}76\,\text{Hz}} = 15.680$$

Debido a que el módulo truncado tiene que ser 32.064, lo más probable es que la inicialización del contador se esté haciendo con un valor erróneo cuando se inicia un nuevo ciclo. El valor de inicialización real se determina de la manera siguiente:

$$
\begin{aligned}
\text{módulo truncado} &= \text{módulo completo} - \text{valor de inicialización} \\
\text{valor de inicialización} &= \text{módulo completo} - \text{módulo truncado} \\
&= 65.536 - 15.680 \\
&= 49.856 \\
&= C2C0_{16}
\end{aligned}
$$

Esto muestra que el valor de inicialización del contador es, cada vez que se inicia un nuevo ciclo, $C2C0_{16}$ en lugar de $82C0_{16}$

La carga de los contadores 1, 2 y 3 se realiza correctamente, pero no la del contador 4. Dado que $C_{16} = 1100_2$, la entrada D_2 del contador 4 está a nivel ALTO cuando debería estar a nivel BAJO. Lo más probable es que este fallo esté causado por una **entrada en circuito abierto**. Hay que comprobar también un circuito abierto externo causado por una mala soldadura en las conexiones, un conductor roto o un pin curvado del circuito integrado. Si no se detecta ninguno de estos fallos, debe reemplazarse el CI y el contador funcionará correctamente.

Problema relacionado Determinar cuál sería la frecuencia de salida en TC 4 si la entrada D_3 del contador 3 estuviera en circuito abierto.

Contadores implementados con flip-flops individuales

Los contadores implementados con flip-flops individuales y circuitos integrados de puertas son, algunas veces, más difíciles de comprobar en caso de fallo, ya que hay muchas más entradas y salidas con conexiones externas, que en un CI contador. La secuencia de un contador se puede alterar por la existencia de un único circuito abierto o un cortocircuito en una entrada o salida, como nos muestra el Ejemplo 8.11.

CONSEJOS PRÁCTICOS

Cuando se observa la relación temporal entre dos señales digitales en un osciloscopio de doble traza, la forma adecuada de disparar el osciloscopio es mediante la señal más lenta de las dos. La razón de esto es que la señal más lenta dispone de menos puntos de disparo que la señal más rápida, por lo que no existirá ambigüedad en el inicio del barrido. El disparo en modo vertical utiliza una señal compuesta de ambos canales y nunca se debería emplear para determinar información temporal absoluta. Dado que, generalmente, las señales de reloj son las más rápidas en un sistema digital, no se deberían utilizar para el disparo.

EJEMPLO 8.11

Supongamos que se observan las formas de onda de salida que se indican, para el contador de la Figura 8.62. Determinar si existe algún problema en el contador.

Solución La forma de onda Q_2 es incorrecta. La forma de onda correcta se indica mediante una línea discontinua. Puede observar que la forma de onda Q_2 tiene exactamente la misma forma que Q_1. Esto indica que la misma señal que está haciendo bascular a FF1 controla también a FF2.

Si comprobamos las entradas J y K de FF2, encontramos una señal que tiene la misma forma que Q_0. Este resultado indica que Q_0 pasa de alguna manera a través de la puerta AND. Esto sólo puede ocurrir si la entrada Q_1 de la puerta AND está siempre a nivel ALTO. Pero acabamos de ver que Q_1 tiene una forma de onda correcta. Esta observación nos conduce a la conclusión de que la entrada inferior de la puerta AND tiene que estar, internamente, en circuito abierto, por lo que actúa como un nivel ALTO. Es necesario entonces reemplazar la puerta AND y volver a comprobar el circuito.

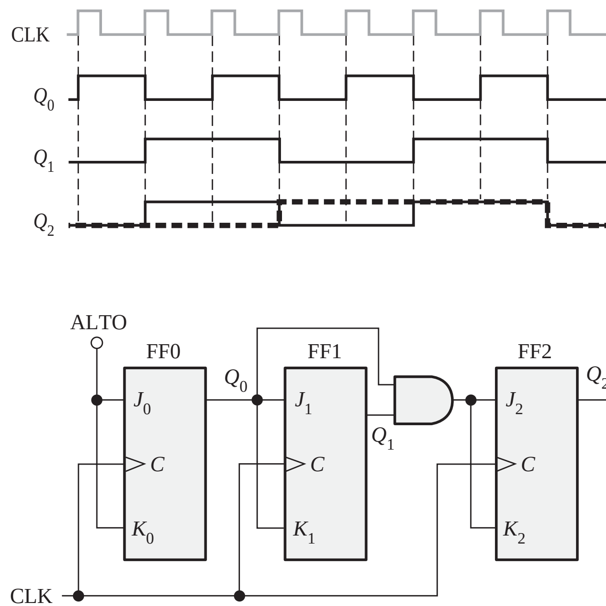


FIGURA 8.62

Problema relacionado Describir la salida Q_2 del contador de la Figura 8.62 si la salida Q_1 de FF1 está en circuito abierto.

REVISIÓN DE LA SECCIÓN 8.9

1. ¿Cuáles son los fallos que pueden causar que el contador de la Figura 8.59 no tenga actividad de impulsos en ninguna de las salidas *TC*?
2. ¿Qué ocurre si el inversor de la Figura 8.61 tiene una salida en circuito abierto?



APLICACIÓN A LOS SISTEMAS DIGITALES

El sistema de control de luces de los semáforos que se ha iniciado en el Capítulo 6 y con el que se ha continuado en el Capítulo 7 se completa en este capítulo. En el Capítulo 6 se ha desarrollado la lógica combinacional.

En el Capítulo 7 se han desarrollado los circuitos de temporización.

En este capítulo se aborda la lógica secuencial y se conectan todos los bloques para generar el sistema de control completo de las luces de los semáforos. De nuevo, el diagrama de bloques global del sistema se muestra en la Figura 8.63.

Requisitos de la lógica secuencial

La lógica secuencial controla el secuenciamiento de las luces de los semáforos basándose en las entradas procedentes de los circuitos de temporización y del sensor de vehículos. La lógica secuencial generará una secuencia de código Gray de 2 bits para los cuatro estados del sistema indicados en la Figura 8.64.

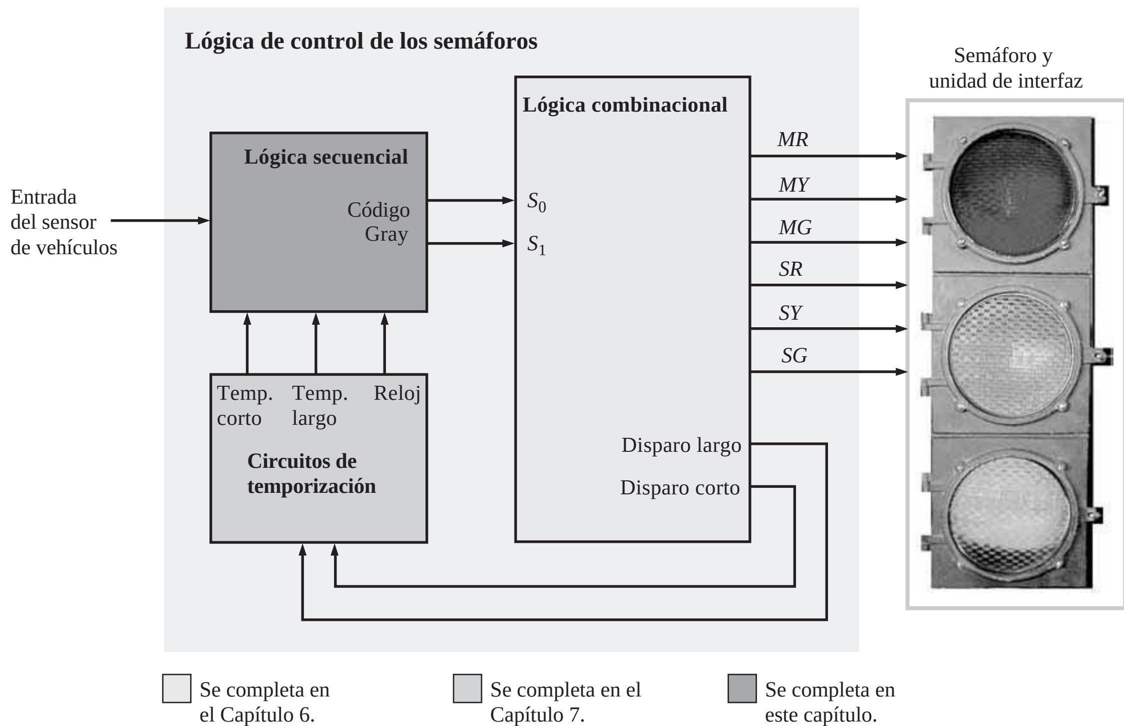


FIGURA 8.63 Diagrama de bloques del sistema de control de semáforos.

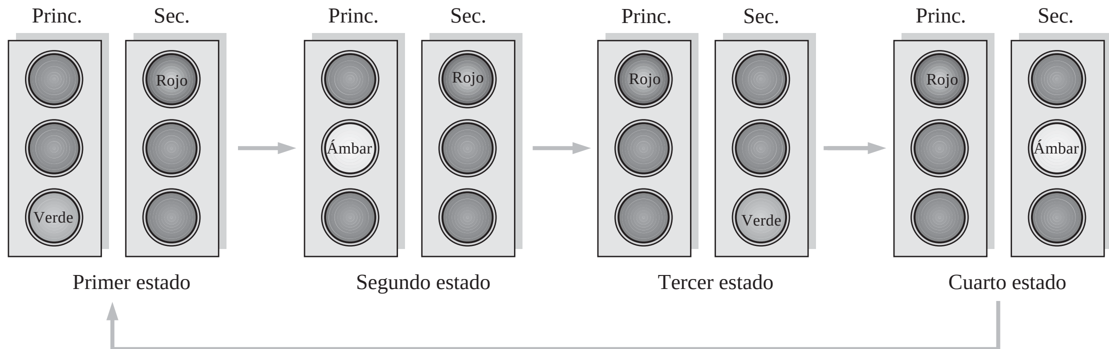


FIGURA 8.64 Secuencia de estados de las luces de los semáforos.

Diagrama de bloques La lógica secuencial consiste en un contador en código Gray de 2 bits y la lógica de entrada asociada, como se muestra en la Figura 8.65.

El contador genera una secuencia de cuatro estados. Las transiciones desde un estado al siguiente están determinadas por el temporizador de 4 s, por el temporizador de 25 s y por la entrada del sensor de vehículos. El reloj del contador es la señal de 10 kHz producida por el oscilador de los circuitos de temporización.

Diagrama de estados El diagrama de estados del sistema se ha introducido en el Capítulo 6 y se muestra de nuevo en la Figura 8.66. En función de este diagrama de estados, se describe a continuación el funcionamiento de la lógica secuencial.

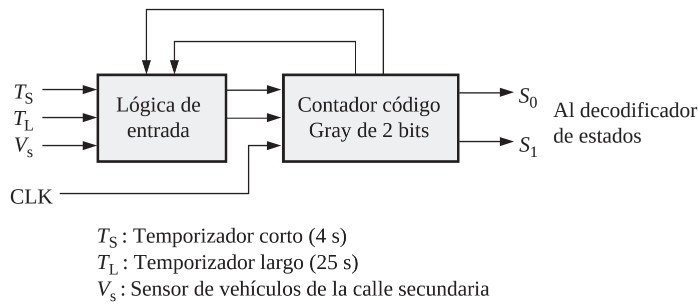


FIGURA 8.65 Diagrama de bloques de la lógica secuencial.

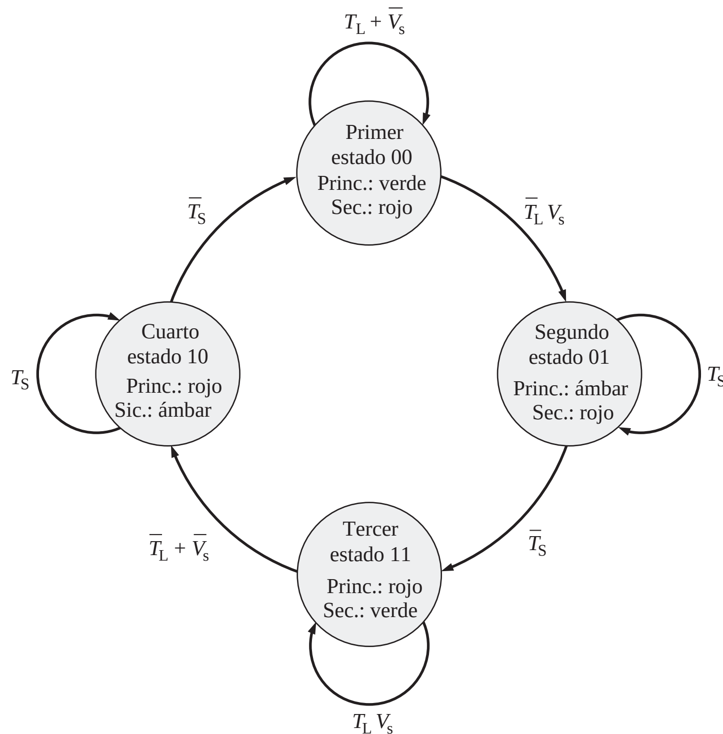


FIGURA 8.66 Diagrama de estados del sistema de control de las luces de los semáforos.

Primer estado El código Gray para este estado es 00. La luz del semáforo de la calle principal está en verde y la de la calle secundaria está en rojo. El sistema permanece en este estado al menos durante 25 s cuando el temporizador largo está *activado* o mientras que no haya vehículos en la calle secundaria. Esto se expresa como $T_L + \overline{V}_S$. El sistema pasa al siguiente estado cuando el temporizador largo está *desactivado* y hay un vehículo en la calle secundaria. Esto se expresa como $(\overline{T}_L V_S)$.

Segundo estado El código Gray para este estado es 01. La luz del semáforo de la calle principal está en ámbar y la de la calle secundaria está en rojo. El sistema permanece en este estado durante 4 s cuando el temporizador corto está activado (T_S) y pasa al siguiente estado cuando este mismo temporizador se desactiva $(\overline{T}_S)$.

Tercer estado El código Gray para este estado es 11. La luz del semáforo de la calle principal está en rojo y la de la

calle secundaria está en verde. El sistema permanece en este estado cuando el temporizador largo está *activado* y hay un vehículo en la calle secundaria. Esto se expresa como $T_L V_S$. El sistema pasa al estado siguiente cuando se *desactiva* el temporizador largo o cuando no hay vehículos en la calle secundaria, lo que se indica como $\overline{T}_L + \overline{V}_S$

Cuarto estado El código Gray para este estado es 10. La luz del semáforo de la calle principal está en rojo y la de la calle secundaria está en ámbar. El sistema permanece en este estado durante 4 s cuando el temporizador corto está *activado* (T_S) y vuelve al primer estado cuando el temporizador corto se desactiva ($\overline{T}_S$).

Implementación de la lógica secuencial El diagrama de la Figura 8.67 muestra que se emplean dos flip-flops D para implementar el contador Gray. Las salidas de la lógica de entrada proporcionan las entradas *D* a los biestables y el contador se sincroniza mediante el reloj de 10 kHz del oscilador. La lógica de entrada tiene cinco variables de entrada: Q_0, Q_1, T_L, T_S y V_S.

En la Tabla 8.13 se muestra la tabla de transiciones del flip-flop D. A partir del diagrama de estados, puede desarrollarse la tabla del estado siguiente, como se muestra en la Tabla 8.14. Las condiciones de entrada para T_L, T_S y V_S para cada combinación de estado actual/estado siguiente se enumeran en la tabla.

Transiciones de salida Q_N		Q_{N+1}	Entradas del flip-flop D
0	⟶	0	0
0	⟶	1	1
1	⟶	0	0
1	⟶	1	1

TABLA 8.13 Tabla de transiciones del flip-flop D.

Estado actual Q_1	Q_0	Estado siguiente Q_1	Q_0	Condiciones de entrada	Entradas FF D_1	D_0
0	0	0	0	$T_L + \overline{V}_s$	0	0
0	0	0	1	$\overline{T}_L V_S$	0	1
0	1	0	1	T_S	0	1
0	1	1	1	$\overline{T}_S$	1	1
1	1	1	1	$T_L V_s$	1	1
1	1	1	0	$\overline{T}_L + \overline{V}_s$	1	0
1	0	1	0	T_S	1	0
1	0	0	0	$\overline{T}_S$	0	0

TABLA 8.14 Tabla del estado siguiente para las transiciones de la lógica secuencial.

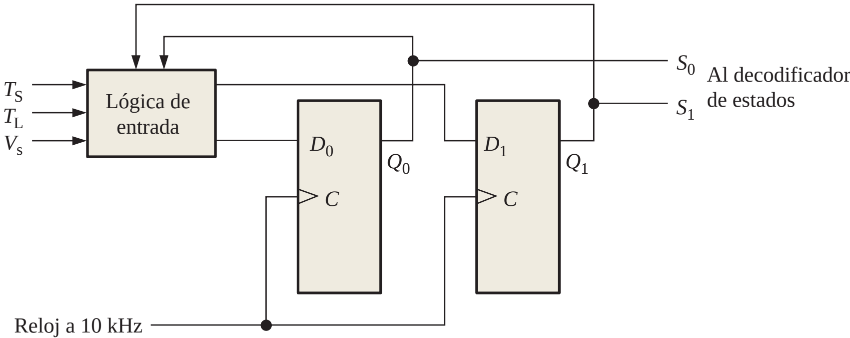


FIGURA 8.67 Diagrama de la lógica secuencial.

A partir de las Tablas 8.13 y 8.14 pueden determinarse las condiciones lógicas requeridas para que cada flip-flop pase al estado 1. Por ejemplo, Q_0 pasa de 0 a 1cuando el estado actual es 00 y la condición de entrada es $\overline{T}_L V_S$, como se indica en la segunda fila de la Tabla 8.13. D_0 debe estar a 1 para hacer que Q_0 pase a 1 o permanezca en 1 en el siguiente impulso de reloj. Para que D_0 sea un 1, puede escribirse una expresión lógica a partir de la Tabla 8.14:

$$\begin{aligned} D_0 &= \overline{Q}_1\overline{Q}_0\overline{T}_L V_s + \overline{Q}_1 Q_0 T_S \\ &+ \overline{Q}_1 Q_0 \overline{T}_S + Q_1 Q_0 T_L V_s \\ &= \overline{Q}_1\overline{Q}_0\overline{T}_L V_s + \overline{Q}_1 Q_0 + Q_1 Q_0 T_L V_s \end{aligned}$$

Puede utilizarse un mapa de Karnaugh para reducir aún más la expresión de D_0

$$D_0 = \overline{Q}_1\overline{T}_L V_s + \overline{Q}_1 Q_0 + Q_0 T_L V_s$$

También, podemos desarrollar la expresión de D_1 a partir de la Tabla 8.14,

$$\begin{aligned} D_1 &= \overline{Q}_1 Q_0 \overline{T}_S + Q_1 Q_0 T_L V_s \\ &+ Q_1 Q_0 \overline{T}_L + Q_1 Q_0 \overline{V}_s + Q_1 \overline{Q}_0 T_S \\ &= \overline{Q}_1 Q_0 \overline{T}_S + Q_1 Q_0 (T_L V_s + \overline{T}_L) \\ &+ Q_1 Q_0 \overline{V}_s + Q_1 \overline{Q}_0 T_S \\ &= \overline{Q}_1 Q_0 \overline{T}_S + Q_1 Q_0 (V_s + \overline{T}_L) \\ &+ Q_1 Q_0 \overline{V}_s + Q_1 \overline{Q}_0 T_S \\ &= \overline{Q}_1 Q_0 \overline{T}_S + Q_1 Q_0 (V_s + \overline{T}_L + \overline{V}_s) \\ &+ Q_1 \overline{Q}_0 T_S \\ &= \overline{Q}_1 Q_0 \overline{T}_S + Q_1 Q_0 + Q_1 \overline{Q}_0 T_S \end{aligned}$$

Puede utilizarse un mapa de Karnaugh para reducir aún más la expresión de D_1

$$D_1 = Q_0 \overline{T}_S + Q_1 T_S$$

D_0 y D_1 se implementan como se muestra en la Figura 8.68.

Combinando la lógica de entrada con el contador de 2 bits, se obtiene el diagrama lógico secuencial completo mostrado en la Figura 8.69.

Sistema de control completo de los semáforos

Ahora que disponemos de los tres bloques (lógica combinacional, circuitos de temporización y lógica secuencial) vamos a combinarlos para formar el sistema completo, cuyo diagrama de bloques es el mostrado en la Figura 8.70.

Circuitos de interfaz Los circuitos de interfaz son necesarios porque la lógica no puede controlar directamente las luces debido a los requisitos de corriente y de tensión. Existen varias formas de proporcionar una interfaz y se proporcionan dos posibles diseños en el Apéndice B.

Práctica de sistemas

- ***Actividad 1*** Utilizar un mapa de Karnaugh para confirmar que la expresión simplificada de D_0 es correcta.
- ***Actividad 2*** Utilizar un mapa de Karnaugh para confirmar que la expresión simplificada de D_1 es correcta.

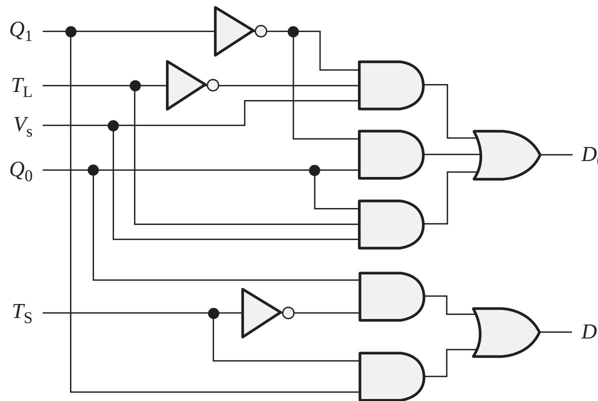


FIGURA 8.68 Lógica de entrada para el contador código Gray de 2 bits.

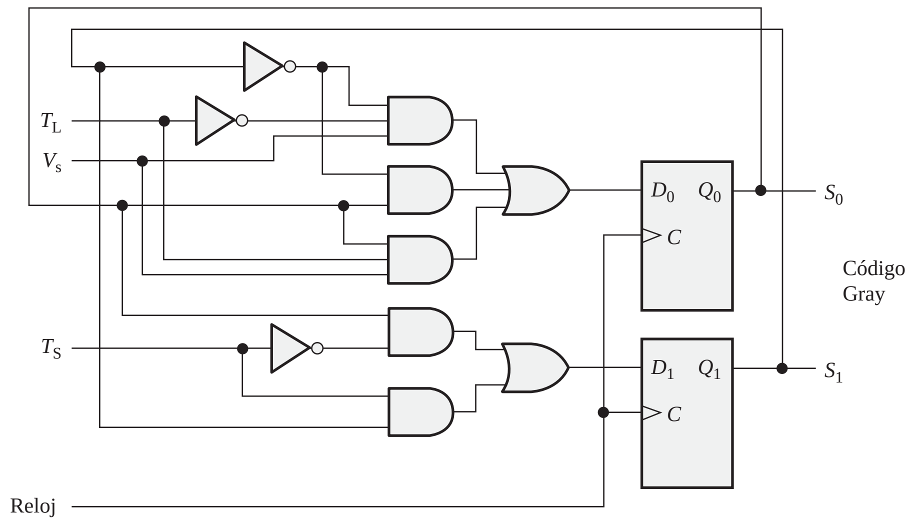


FIGURA 8.69 La lógica secuencial.

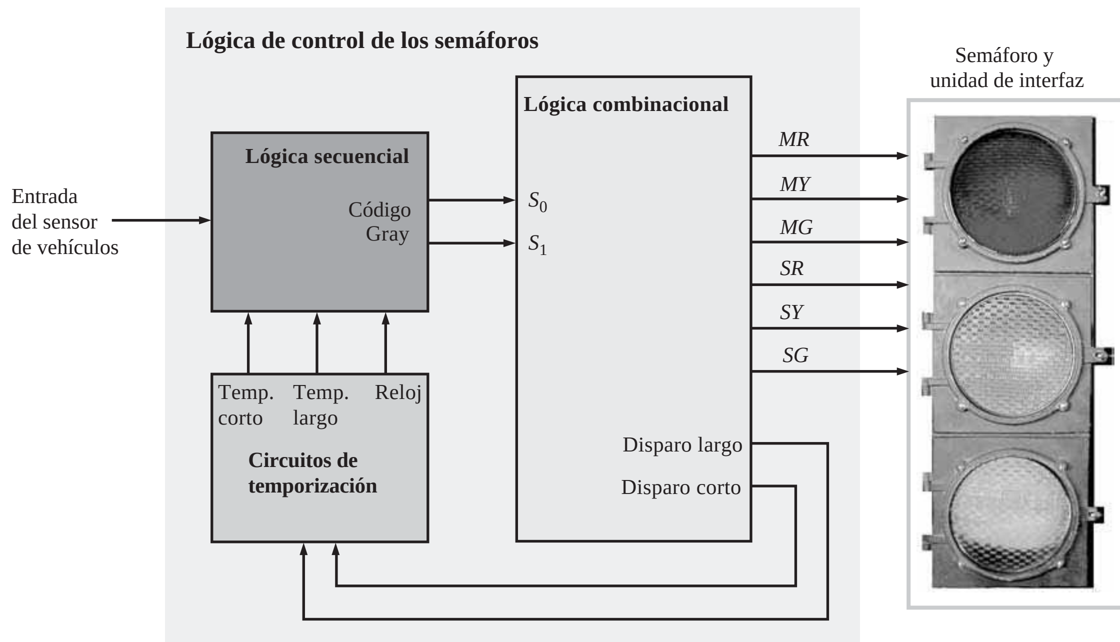


FIGURA 8.70 Diagrama de bloques del sistema de control de luces de los semáforos.

RESUMEN

- Como se muestra en la Figura 8.71, los contadores síncronos y asíncronos únicamente se diferencian en la forma en que se les aplica la señal de reloj. Los contadores síncronos pueden trabajar a frecuencias de reloj mayores que los contadores asíncronos.

Asíncrono

Síncrono

FIGURA 8.71 Comparación de los contadores síncronos y asíncronos.

- En la Figura 8.72 se muestran las conexiones de los circuitos integrados contadores presentados en este capítulo.

Contador binario asíncrono de 4 bits 74LS93

Contador de décadas BCD síncrono con entrada de borrado asíncrona 74F162

Contador binario síncrono de 4 bits con entrada de borrado asíncrona 74HC161

Contador binario síncrono de 4 bits con entrada de borrado síncrona 74HC163

Contador de décadas ascendente/descendente síncrono 74HC190 (*G* es la activación de cuenta)

FIGURA 8.72 Observe que las etiquetas (nombres de las entradas y de las salidas) son consecuentes con el texto, pero pueden diferir con respecto al catálogo del fabricante que esté utilizando. Los dispositivos mostrados son funcionalmente iguales y compatibles en cuanto a pines con los dispositivos del mismo tipo disponibles en otras familias CMOS y TTL de circuitos integrados.

- El módulo máximo de un contador es el número máximo de posibles estados y es función del número de etapas (flip-flops). Por tanto,

  Módulo máximo = 2^n

  donde *n* es el número de etapas del contador. El módulo de un contador es el número *real* de estados de la secuencia, y puede ser igual o menor que el módulo máximo.

- El módulo global de los contadores en cascada es igual al producto de los módulos de los contadores individuales.

PALABRAS CLAVE

Las palabras clave y otros términos que se han resaltado en negrita se encuentran en el glosario final del libro.

Asíncrono Que no ocurre a un mismo tiempo.

Cascada Conectar dispositivos "uno tras otro", como cuando se conectan varios contadores de forma que la salida de un contador esté conectada a la entrada de habilitación del siguiente contador.

Década Que se caracteriza por diez estados o valores.

Diagrama de estados Una descripción gráfica de una secuencia de estados o valores.

Inicio de un nuevo ciclo Sufrir una transición (como ocurre en los contadores) desde el estado final o terminal hasta el estado inicial.

Máquina de estados Sistema lógico que exhibe una secuencia de estados condicionada por la lógica interna y las entradas externas. Cualquier circuito secuencial que exhibe una determinada secuencia de estados.

Módulo El número de estados exclusivos a través de los cuales pasa la secuencia de un contador.

Síncrono Que ocurre de forma simultánea

Valor de fin de cuenta Estado final de la secuencia de un contador.

AUTOTEST

Las respuestas se encuentran al final del capítulo.

1. Los contadores asíncronos se conocen como:

(a) contadores con propagación **(b)** contadores de reloj múltiple

(c) contadores de décadas **(d)** contadores de módulo

2. Un contador asíncrono se diferencia de un contador síncrono en:

(a) el número de estados de su secuencia

(b) el método de sincronización con la señal de reloj

(c) el tipo de flip-flops utilizados

(d) el valor del módulo

3. El módulo de un contador es:

(a) el número de flip-flops

(b) el número real de estados en su secuencia

(c) el número de veces que inicia un nuevo ciclo por segundo

(d) el máximo número posible de estados

4. Un contador binario de 3 bits tiene un módulo máximo de:

(a) 3 **(b)** 6 **(c)** 8 **(d)** 16

5. Un contador binario de 4 bits tiene un módulo máximo de:

(a) 16 **(b)** 32 **(c)** 8 **(d)** 4

6. Un contador de módulo 12 tiene:

(a) 12 flip-flops **(b)** 3 flip-flops

(c) 4 flip-flops **(d)** temporización síncrona

7. ¿Cuál de los siguientes contadores es un ejemplo de un contador con un módulo truncado?
   (a) módulo 8 **(b)** módulo 14
   (c) módulo 16 **(d)** módulo 32
8. Un contador asíncrono de 4 bits está formado por flip-flops que tienen un retardo de propagación de la señal de reloj a Q de 12 ns. ¿Cuánto tiempo tarda el contador en iniciar un nuevo ciclo desde 1111 a 0000?
9. Un contador BCD es un ejemplo de
   (a) contador de módulo completo
   (b) un contador de décadas
   (c) un contador de módulo truncado
   (d) las respuestas (b) y (c)
10. En un contador BCD 8421, ¿cuál de los siguientes estados es un estado no válido?
   (a) 1100 **(b)** 0010 **(c)** 0101 **(d)** 1000
11. Tres contadores de módulo 10 en cascada tienen un módulo global de:
   (a) 30 **(b)** 100 **(c)** 1000 **(d)** 10.000
12. Se aplica una frecuencia de reloj de 10 MHz a un contador en cascada formado por un contador de módulo 5, un contador de módulo 8 y dos contadores de módulo 10. La frecuencia de salida más baja posible es:
   (a) 10 kHz **(b)** 2,5 kHz **(c)** 5 kHz **(d)** 25 kHz
13. Un contador ascendente/descendente de 4 bits se encuentra en estado binario cero. El siguiente estado en el modo descendente es:
   (a) 0001 **(b)** 1111 **(c)** 1000 **(d)** 1110
14. El valor fin de cuenta de un contador binario de módulo 13 es:
   (a) 0000 **(b)** 1111 **(c)** 1101 **(d)** 1100

PROBLEMAS

Las respuestas a los problemas impares se encuentran al final del libro.

SECCIÓN 8.1. Funcionamiento del contador asíncrono

1. Para el contador asíncrono de la Figura 8.73, dibujar el diagrama de tiempos completo para ocho impulsos de reloj, indicando las formas de onda de la señal de reloj, de Q_0 y de Q_1.

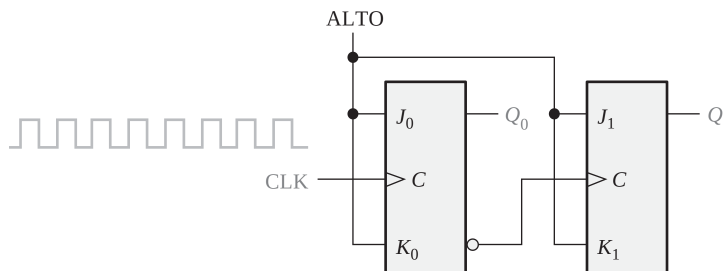


FIGURA 8.73

2. Para el contador asíncrono de la Figura 8.74, dibujar el diagrama de tiempos completo para dieciséis impulsos de reloj, indicando las formas de onda de la señal de reloj, Q_0, Q_1 y Q_2.
3. En el contador del Problema 2, suponer que cada flip-flop tiene un retardo de propagación, entre el impulso de disparo de reloj y el cambio en la salida Q, de 8 ns. Determinar el retardo

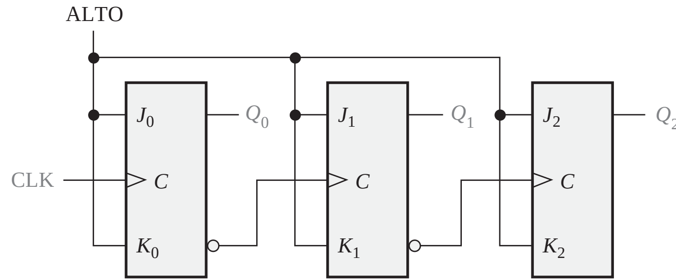


FIGURA 8.74

en el caso peor (el más largo) a partir de que se aplica un impulso de reloj hasta que el contador alcanza un determinado estado. Especificar el estado o estados para los que se produce dicho caso peor.

4. Indicar cómo se conecta un contador asíncrono de 4 bits 74LS93, para obtener cada uno de los siguientes módulos:

(a) 9 **(b)** 11 **(c)** 13 **(d)** 14 **(e)** 15

SECCIÓN 8.2 Funcionamiento del contador síncrono

5. Si el contador del Problema 3 fuera síncrono en lugar de asíncrono, ¿cuál sería el retardo más largo?

6. Dibujar el diagrama de tiempos completo para el contador binario síncrono de cinco etapas de la Figura 8.75. Verificar que las formas de onda de las salidas *Q* representan el número binario correcto después de cada impulso de reloj.

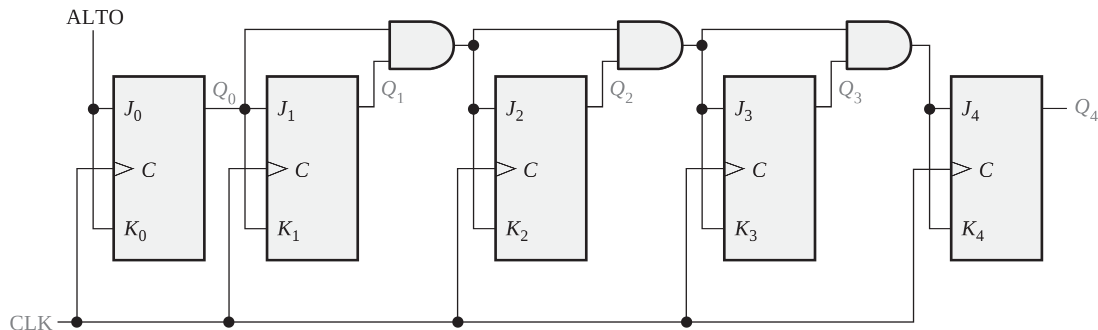


FIGURA 8.75

7. Analizando las entradas *J* y *K* de cada flip-flop antes de cada impulso de reloj, probar que el contador de décadas de la Figura 8.76 avanza a través de una secuencia BCD. Explicar, en cada caso, cómo estas condiciones hacen que el contador pase al siguiente estado correcto.

8. Las formas de onda de la Figura 8.77 se aplican a las entradas de habilitación, borrado y de reloj , como se indica. Dibujar las señales de salida del contador en función de estas entradas. La entrada de borrado es asíncrona.

9. En la Figura 8.78 se muestra un contador de décadas BCD. Se aplican las entradas de reloj y de borrado que se indican. Determinar las formas de onda de las salidas del contador (Q_0, Q_1, Q_2 y Q_3). La entrada de borrado es síncrona y el contador, inicialmente, está en el estado binario 1000.

10. Las formas de onda de la Figura 8.79 se aplican a un contador 74HC163. Determinar las salidas *Q* y *RCO*. Las entradas son $D_0 = 1$, $D_1 = 1$, $D_2 = 0$ y $D_3 = 1$.

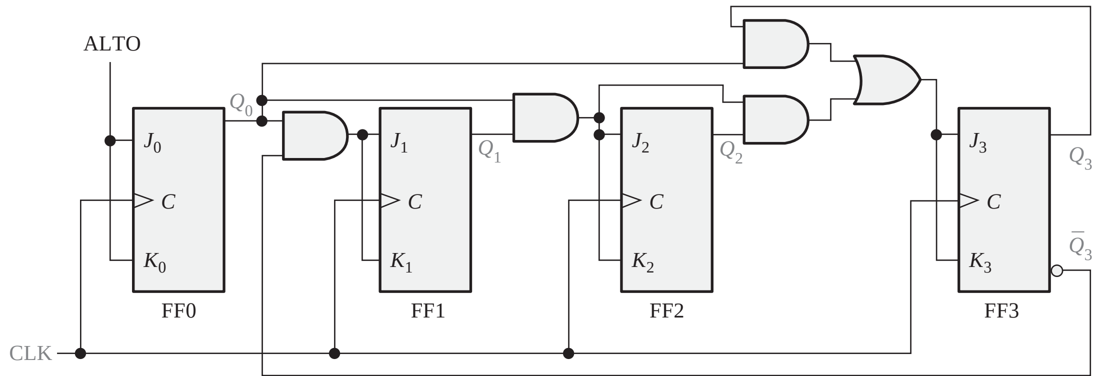


FIGURA 8.76

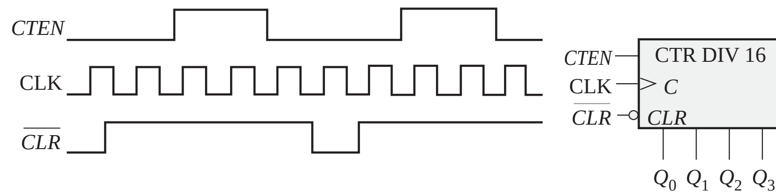


FIGURA 8.77

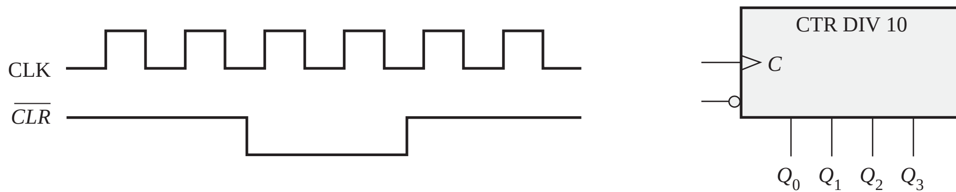


FIGURA 8.78

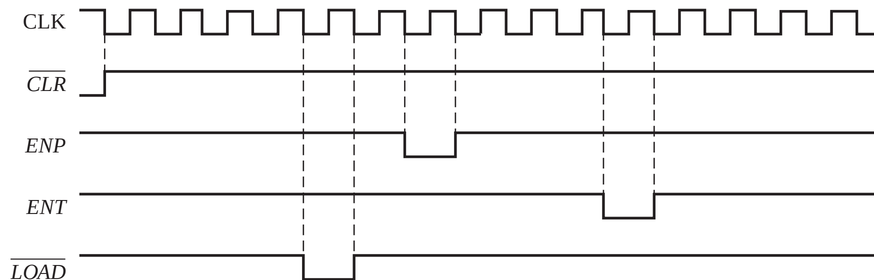


FIGURA 8.79

11. Las formas de onda de la Figura 8.79 se aplican a un contador 74F162. Determinar las salidas Q y TC. Las entradas son $D_0 = 1$, $D_1 = 0$, $D_2 = 0$ y $D_3 = 1$.

SECCIÓN 8.3. Contadores ascendentes/descendentes síncronos

12. Dibujar un diagrama de tiempos completo para un contador ascendente/descendente de 3 bits que sigue la siguiente secuencia. Indicar cuándo el contador está en modo ascendente y cuándo está en modo descendente. Suponer que es disparado por flanco positivo.

0, 1, 2, 3, 2, 1, 2, 3, 4, 5, 6, 5, 4, 3, 2, 1, 0

13. Dibujar la forma de onda de salida Q de un contador ascendente/descendente 74HC190 con las formas de onda de entrada mostradas en la Figura 8.80. Las entradas de datos están a cero. Comenzar la cuenta en el estado 0000.

FIGURA 8.80

SECCIÓN 8.4 Diseño de los contadores síncronos

14. Determinar la secuencia del contador de la Figura 8.81.

FIGURA 8.81

15. Determinar la secuencia del contador de la Figura 8.82. Comenzar con el contador borrado.

FIGURA 8.82

16. Diseñar un contador que genere la siguiente secuencia. Utilizar flip-flops J-K.
00, 10, 01, 11, 00 ...

17. Diseñar un contador que genere la siguiente secuencia binaria. Utilizar flip-flops J-K.
1, 4, 3, 5, 7, 6, 2, 1 ...

18. Diseñar un contador que genere la siguiente secuencia binaria. Utilizar flip-flops J-K.
0, 9, 1, 8, 2, 7, 3, 6, 4, 5, 0, ...

19. Diseñar un contador binario que genere la secuencia que indica el diagrama de estados de la Figura 8.83.

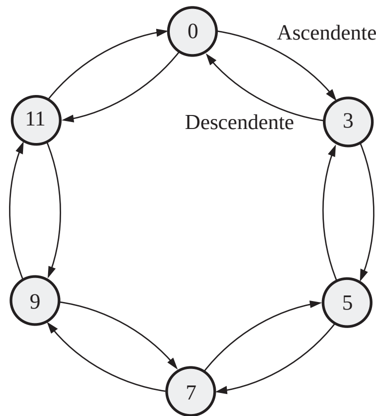


FIGURA 8.83

SECCIÓN 8.5 Contadores en cascada

20. Para cada una de las configuraciones en cascada de la Figura 8.84, determinar la frecuencia de la señal en cada punto señalado con un número encerrado en un círculo, y calcular los módulos globales.

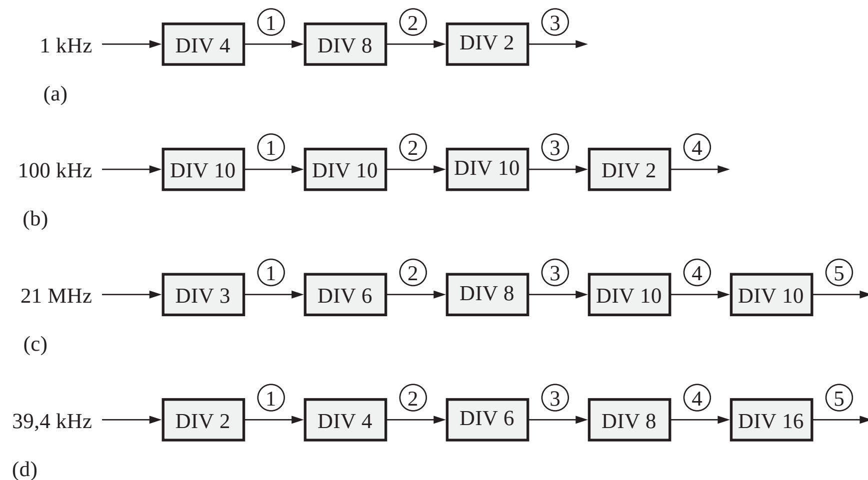


FIGURA 8.84

21. Ampliar el contador de la Figura 8.41 para crear un contador divisor por 10.000 y un contador divisor por 100.000.

22. Por medio de diagramas de bloques generales, indicar cómo se obtendrán las siguientes frecuencias a partir de una señal de reloj a 10 MHz, empleando flip-flops, contadores de módulo 5 y contadores de décadas.

(a) 5 MHz **(b)** 2,5 MHz **(c)** 2 MHz
(d) 1 MHz **(e)** 500 kHz **(f)** 250 kHz
(g) 62,5 kHz **(h)** 40 kHz **(i)** 10 kHz **(j)** 1 kHz

SECCIÓN 8.6 Decodificación de los contadores

23. Dado un codificador de décadas BCD con sólo disponibles las salidas *Q*, definir la lógica requerida para decodificar cada uno de los estados futuros e indicar cómo se conectaría al contador. Se precisa una salida a nivel ALTO para indicar cada estado decodificado. El MSB es el de la izquierda.

(a) 0001

(b) 0011

(c) 0101

(d) 0111

(e) 1000

24. Para el contador binario de 4 bits conectado al decodificador de la Figura 8.85, determinar cada forma de onda de salida del decodificador en función de los impulsos de reloj.

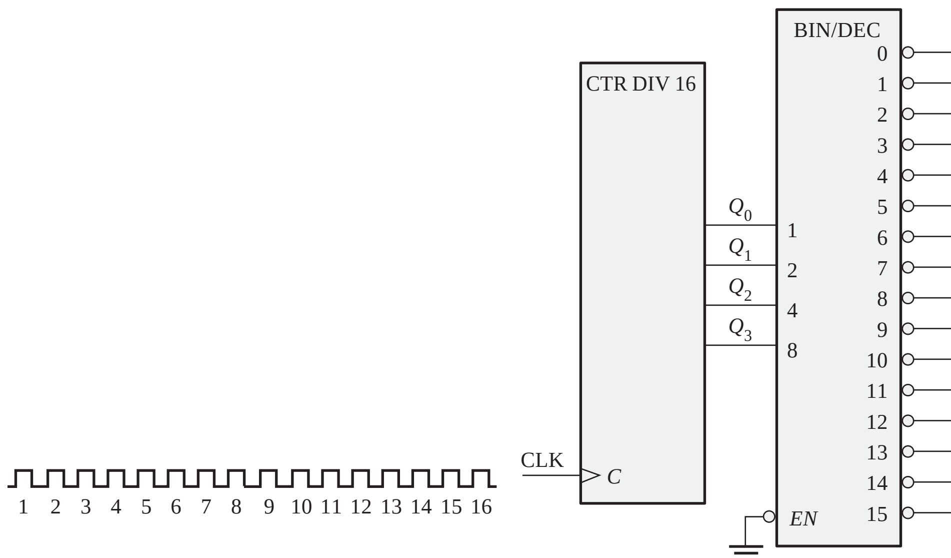


FIGURA 8.85

25. Si el contador de la Figura 8.85 es asíncrono, determinar dónde se producen los glitches en las señales de salida del decodificador.

26. Modificar el circuito de la Figura 8.85 para eliminar los *glitches* de decodificación.

27. Analizar la ocurrencia de *glitches* en la salida de la puerta de decodificación en el contador de la Figura 8.45. Si se producen *glitches*, sugerir una forma de eliminarlos.

28. Analizar la ocurrencia de *glitches* en las salidas de las puertas de decodificación en el contador de la Figura 8.46. Si éstos se producen, modificar el diseño para eliminarlos.

SECCIÓN 8.7 Aplicaciones de los contadores

29. Suponer que el reloj digital de la Figura 8.51 se inicializa a las doce horas. Determinar el estado binario de cada contador después de que se hayan producido sesenta y dos impulsos de 60 Hz de frecuencia.

30. ¿Cuál es la frecuencia de salida de cada contador en el circuito del reloj digital de la Figura 8.51?

31. Para el sistema de control del aparcamiento de coches de la Figura 8.54, en la Figura 8.86 se presenta una secuencia patrón de entrada y los impulsos del sensor para un determinado período de 24 horas. Si ya había 53 coches en el garaje al inicio del período, ¿cuál es el estado del contador pasadas las 24 horas?

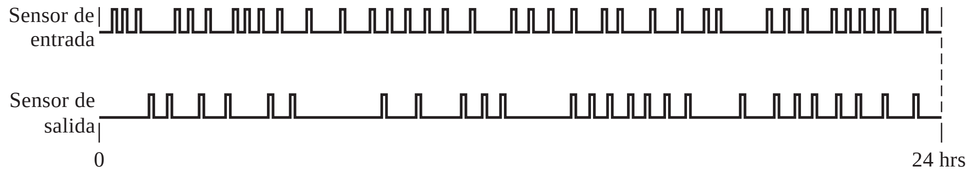


FIGURA 8.86

32. El número binario correspondiente al decimal 57 se presenta en las entradas de datos en paralelo del convertidor paralelo-serie de la Figura 8.56 (D_0 es el LSB). Inicialmente, el contador contiene todo ceros y se aplica una señal de reloj a 10 kHz. Desarrollar el diagrama de tiempos que muestre el reloj, las salidas del contador y la salida de datos serie.

SECCIÓN 8.9 Localización de averías

33. Para el contador de la Figura 8.1, dibujar el diagrama de tiempos para las formas de onda Q_0 y Q_1 si se produce alguno de los fallos siguientes (suponer que, inicialmente, Q_0 y Q_1 están a nivel BAJO):

(a) la entrada de reloj de FF0 está cortocircuitada a masa.

(b) la salida Q_0 está en circuito abierto.

(c) la entrada de reloj de FF1 está en circuito abierto

(d) la entrada *J* de FF0 está en circuito abierto

(e) la entrada *K* de FF1 está cortocircuitada a masa.

34. Resolver el Problema 33 para el contador de la Figura 8.11.

35. Aislar el fallo del contador de la Figura 8.3, analizando las formas de onda de la Figura 8.87.

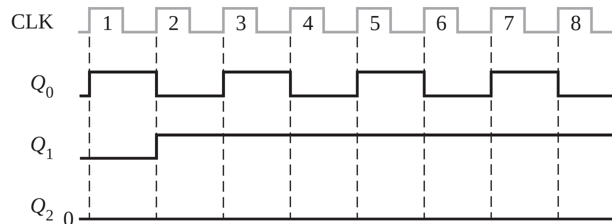


FIGURA 8.87

36. A partir del diagrama de señales de la Figura 8.88, determinar el fallo más probable en el contador de la Figura 8.14.

37. Resolver el Problema 36, si la salida Q_2 se corresponde con la forma de onda de la Figura 8.89. Las salidas Q_0 y Q_1 son las de la Figura 8.88.

38. Se aplica una señal de reloj de 5 MHz al contador en cascada de la Figura 8.44 y se mide una frecuencia de 76,2939 Hz en la última salida *RCO*. ¿Es esto correcto? Si no lo es, ¿cuál es el fallo más probable?

39. Desarrollar una tabla para probar el contador de la Figura 8.44, que muestre la frecuencia de la última salida *RCO*, para todos los posibles fallos que se producen cuando cada una de las entradas de datos (D_0, D_1, D_2 y D_3) está en circuito abierto. Utilizar una frecuencia de prueba de reloj de 10 MHz.

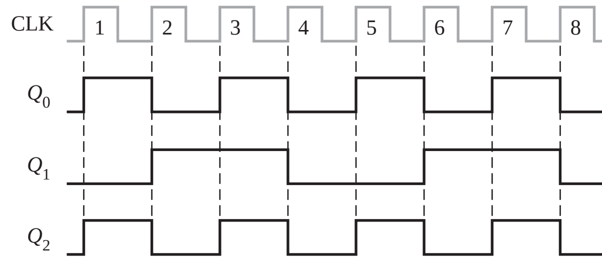


FIGURA 8.88

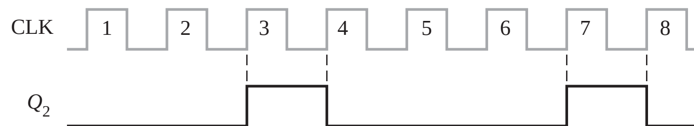


FIGURA 8.89

40. El display de 7-segmentos para las decenas de horas del sistema de reloj digital de la Figura 8.51 presenta continuamente en el display un 1. Los restantes dígitos funcionan correctamente. ¿Cuál es el problema?

41. ¿Cuál sería la indicación visual si la salida Q_1 del contador de decenas de minutos de la Figura 8.51 estuviera en circuito abierto? Consultar también la Figura 8.52.

42. Un determinado día (posiblemente un lunes) los dueños de las plazas del garaje que tiene el sistema de control descrito en las Figuras 8.54 y 8.55, comienzan a presentar quejas. Los dueños dicen que pueden entrar en el garaje porque la barrera está levantada y el cartel de COMPLETO está apagado, pero que una vez que han entrado no pueden encontrar una plaza vacía. Si fuera el técnico encargado de mantener este sistema, ¿cuál pensaría que era el problema? ¿Cómo localizaría la avería y repararía el sistema lo más rápidamente posible?



Aplicación a los sistemas digitales

43. Implementar la lógica de entrada del circuito secuencial del sistema de control de semáforos utilizando sólo puertas NAND.

44. Reemplazar los flip-flops D del contador en código Gray de dos bits de la Figura 8.67 por flip-flops J-K.

45. Especificar cómo se cambiaría el intervalo de la luz verde de 25 s a 60 s.



Problemas especiales de diseño

46. Diseñar un contador de módulo 1000, utilizando contadores de décadas 74F162.

47. Modificar el diseño del contador de la Figura 8.44 para conseguir un módulo de 30.000.

48. Repetir el Problema 47 para obtener un módulo de 50.000.

49. Modificar el reloj digital de las Figuras 8.51, 8.52 y 8.53 para que pueda reinicializarse en cualquier instante.

50. Diseñar un circuito de alarma para que el reloj digital pueda detectar un intervalo de tiempo predeterminado (horas y minutos únicamente) y generar un señal que active una alarma audible.

51. Modificar el diseño del circuito de la Figura 8.55 para 1000 y 3000 plazas de garaje.

52. Implementar la lógica de conversión de datos paralelo-serie de la Figura 8.56 con dispositivos de función fija específicos.

53. En el Problema 15, se ha determinado que el contador entra en un bucle y alterna entre dos estados. Esto sucede como resultado de un fallo de diseño. Diseñar de nuevo el contador para que cuando entre en el segundo de los estados del bucle, se inicie un nuevo ciclo en el estado de todo ceros con el siguiente impulso de reloj.

54. Modificar el diagrama de bloques del sistema de control de semáforos de la Figura 8.63, para añadir una señal de giro a la izquierda durante 15 segundos en la calle principal, inmediatamente antes de la luz verde.

RESPUESTAS

REVISIONES DE CADA SECCIÓN

SECCIÓN 8.1. Funcionamiento del contador asíncrono

1. Asíncrono significa que cada flip-flop posterior al primero se activa por medio de la salida del flip-flop precedente.
2. Un contador de módulo 14 tiene catorce estados, requiriéndose cuatro flip-flops.

SECCIÓN 8.2 Funcionamiento del contador síncrono

1. Todos los flip-flops de un contador síncrono se sincronizan simultáneamente con la señal de reloj.
2. El contador se puede inicializar en cualquier estado.
3. El contador se activa cuando *ENP* y *ENT* están a nivel ALTO; *RCO* pasa a nivel ALTO cuando se alcanza el estado final de la secuencia.

SECCIÓN 8.3. Contadores ascendentes/descendentes síncronos

1. El contador pasa al estado 1001
2. ASCENDENTE: 1111, DESCENDENTE: 0000; el siguiente estado es 1111.

SECCIÓN 8.4 Diseño de los contadores síncronos

1. $J = 1$, $K = \text{X}$ (indiferente)
2. $J = X$ (indiferente), $K = 0$
3. **(a)** El estado siguiente es 1011

   (b) Q_3 (MSB): modo no cambio o SET; Q_2: modo no cambio o RESET; Q_1: modo no cambio o SET; Q_0 (LSB): modo SET o de basculación.

SECCIÓN 8.5 Contadores en cascada

1. Tres contadores de décadas producen ÷ 1000, cuatro contadores de décadas producen ÷10.000.
2. **(a)** ÷20: flip-flop y divisor por 10

   (b) ÷32: flip-flop y divisor por 16

   (c) ÷160: divisor por 16 y divisor por 10

   (d) ÷320: divisor por 16, divisor por 10 y flip-flop.

SECCIÓN 8.6 Decodificación de los contadores

1. **(a)** No hay ningún estado transitorio, porque hay un único cambio de bit.

   (b) 0000, 0001, 0010, 0101, 0110, 0111

   (c) No hay ningún estado transitorio, porque hay un único cambio de bit.

   (d) 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001, 1010, 1011, 1100, 1101, 1110

SECCIÓN 8.7 Aplicaciones de los contadores

1. La puerta G_1 pone a cero el flip-flop en el primer impulso de reloj después de alcanzar el valor 12. La puerta G_2 decodifica el valor 12 para inicializar el contador a 0001.
2. El contador de décadas para las horas avanza a través de cada estado desde cero hasta nueve y, al pasar de nueve a cero para iniciar un nuevo ciclo, el flip-flop bascula al estado SET. Esto da lugar a que se presente un 10 en el display. Cuando el contador de décadas para las horas está en el estado 12, las puertas de decodificación NAND hacen que el contador inicie un nuevo ciclo en el estado 1 con el siguiente impulso de reloj. El flip-flop pasa a estado RESET. Esto hace que aparezca un 1 (01) en el display.

SECCIÓN 8.8 Símbolos lógicos con notación de dependencia

1. *C*: control, usualmente reloj; *M*: modo; *G*: AND
2. *D* indica almacenamiento de datos

SECCIÓN 8.9 Localización de averías

1. No hay impulsos en las salidas *TC*: *CTEN* del primer contador está cortocircuitada a masa o a un nivel BAJO; la entrada de reloj del primer contador está en circuito abierto; la línea de reloj está cortocircuitada a masa o a un nivel BAJO; la salida *TC* del primer contador está cortocircuitada a masa o a un nivel BAJO.
2. Con la salida del inversor en circuito abierto, el contador no puede comenzar un nuevo ciclo en el valor de carga predeterminado, sino que actúa como un contador de módulo completo.

PROBLEMAS RELACIONADOS

8.1 *Véase* la Figura 8.90.

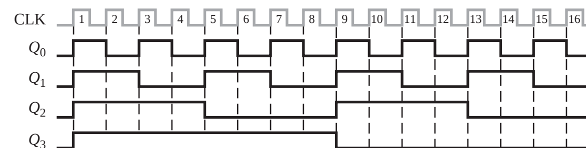


FIGURA 8.90

8.2 Conectar Q_0 a la puerta NAND como tercera entrada (Q_2 y Q_3 son dos de las entradas). Conectar la línea $\overline{CLR}$ a la entrada $\overline{CLR}$ de FF0, así como de FF2 y FF3.

8.3 *Véase* la Figura 8.91.

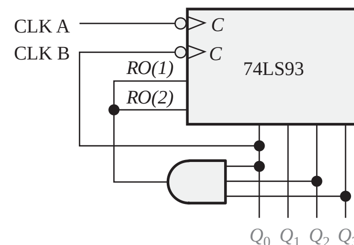


FIGURA 8.91

8.4 *Véase* la Figura 8.92.

8.5 *Véase* la Tabla 8.15.

8.6 La aplicación del álgebra de Boole a la lógica de la Figura 8.37 demuestra que la salida de cada puerta OR está de acuerdo con la expresión del paso 5.

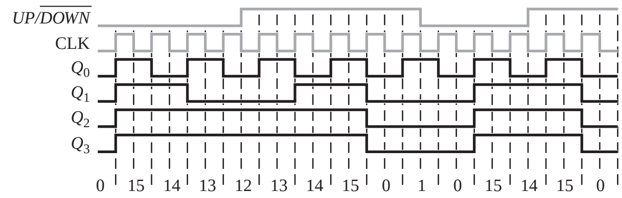


FIGURA 8.92

Estado actual no válido			Entradas *J-K*						Estado siguiente		
Q_2	Q_1	Q_0	J_2	K_2	J_1	K_1	J_0	K_0	Q_2	Q_1	Q_0
0	0	0	0	0	1	1	1	1	0	1	1
0	1	1	1	1	1	1	1	1	1	0	0
1	0	0	0	0	1	1	1	0	1	1	1 estado válido
1	1	0	1	1	1	1	1	0	0	0	1 estado válido

TABLA 8.15

8.7 Se requieren contadores de cinco décadas. 10^5 = 100.000

8.8 f_{Q0} = 1 MHz/[(10)(2)] = 50 kHz

8.9 *Véase* la Figura 8.93.

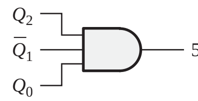


FIGURA 8.93

8.10 Debería cargarse $8AC0_{16}$. $16^4 - 8AC0_{16} = 65.536 - 32.520 = 30.016$

$f_{TC4} = 10$ MHz/30,016 = 332,2 Hz

8.11 *Véase* la Figura 8.94.

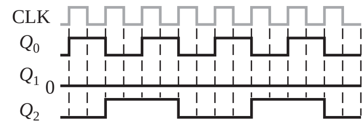


FIGURA 8.94

AUTOTEST

1. (a) **2.** (b) **3.** (b) **4.** (c) **5.** (a)
6. (c) **7.** (b) **8.** (c) **9.** (d) **10.** (a)
11. (c) **12.** (b) **13.** (b) **14.** (d)

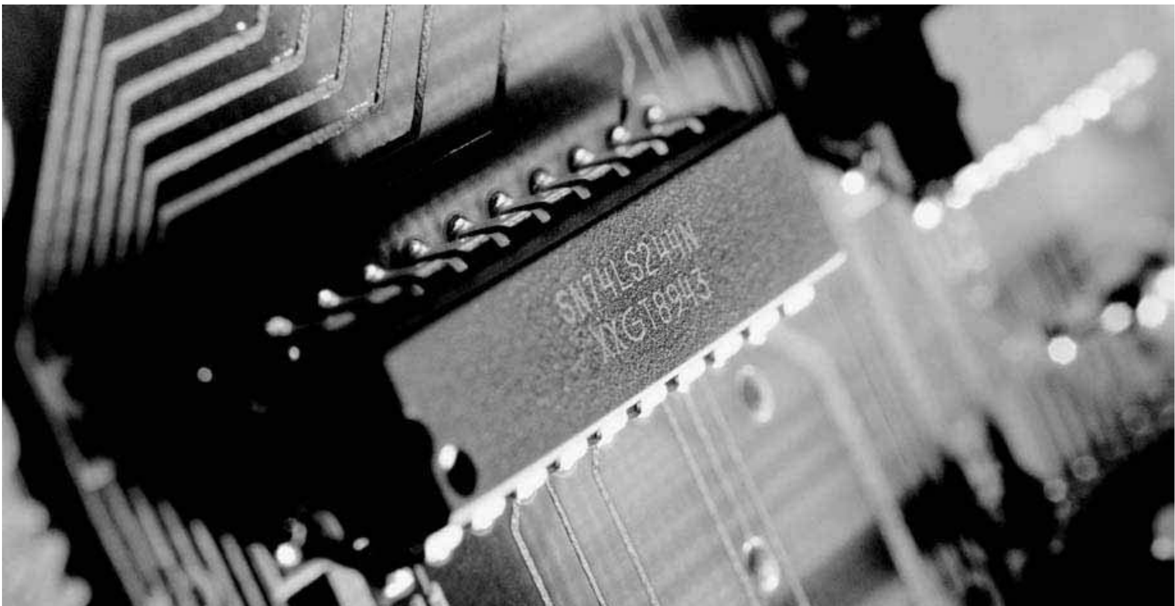
9

REGISTROS DE DESPLAZAMIENTO

CONTENIDO DEL CAPÍTULO

OBJETIVOS DEL CAPÍTULO

- Identificar las formas básicas de movimiento de datos en los registros de desplazamiento.
- Explicar cómo funcionan los registros de desplazamiento con: entrada y salida serie, entrada serie



y salida paralelo, entrada paralelo y salida serie, entrada y salida paralelo.

- Describir cómo funciona un registro de desplazamiento bidireccional.
- Determinar la secuencia de un contador Johnson.
- Configurar un contador en anillo para generar una secuencia específica.
- Realizar un contador en anillo a partir de un registro de desplazamiento.
- Utilizar un registro de desplazamiento para implementar un convertidor de datos serie-paralelo.
- Implementar un codificador de teclado básico controlado por un registro de desplazamiento.
- Interpretar los símbolos estándar ANSI/IEEE 91-1984 de los registros de desplazamiento con notación de dependencia.
- Utilizar los registros de desplazamiento en un sistema digital.

PALABRAS CLAVE

- Registro
- Etapa
- Desplazamiento
- Carga
- Bidireccional

INTRODUCCIÓN

Los registros de desplazamiento son un tipo de circuitos lógicos secuenciales, que están íntimamente relacionados con los contadores digitales. Los registros se utilizan principalmente para almacenar datos digitales y, normalmente, no poseen una secuencia característica interna de estados como los contadores. Sin embargo, existen excepciones, que abordaremos en la Sección 9.7.

En este capítulo, se estudian los tipos básicos de registros de desplazamiento y se presentan diversas aplicaciones. También se introduce un importante método para la localización de averías.

DISPOSITIVOS LÓGICOS DE FUNCIÓN FIJA

74XX164	74XX165	74XX174
74XX194	74XX195	

■■■ APLICACIÓN A LOS SISTEMAS DIGITALES

Esta aplicación sobre sistemas digitales ilustra los conceptos que se aprenderán en este capítulo. Se presenta un sistema de seguridad para acceso, que controla las alarmas de un edificio. Este sistema utiliza dos tipos de registros de desplazamiento, así como otros dispositivos que se han visto en los capítulos anteriores. El sistema también incluye una memoria, dispositivo que se tratará en la sección de aplicaciones a los sistemas digitales del Capítulo 10.

9.1 FUNCIONES BÁSICAS DE LOS REGISTROS DE DESPLAZAMIENTO

Los registros de desplazamiento están formados por un conjunto de flip-flops, y son muy importantes en las aplicaciones que precisan almacenar y transferir datos dentro de un sistema digital. La diferencia básica entre un registro y un contador es que un registro no tiene una secuencia de estados específica, excepto en ciertas aplicaciones muy especializadas. En general, un registro se utiliza únicamente para almacenar y desplazar datos (1s y 0s), que introduce en él una fuente externa y, normalmente, no posee ninguna secuencia característica interna de estados.

Al finalizar esta sección, el lector deberá ser capaz de :

■ Explicar cómo un flip-flop almacena un bit de datos. ■ Definir la capacidad de almacenamiento de un registro de desplazamiento. ■ Definir la capacidad de desplazamiento de un registro

▲ *Un registro puede estar formado por uno o más flip-flops que se utilizan para almacenar y desplazar datos.*

Un ***registro*** es un circuito digital con dos funciones básicas: almacenamiento de datos y movimiento de datos. La capacidad de almacenamiento de un registro le convierte en un tipo importante de dispositivo de memoria. La Figura 9.1 ilustra el concepto de almacenamiento de un 1 o un 0 en un flip-flop D. Como se muestra, se aplica un 1 a la entrada de datos y un impulso de reloj que hace que se almacene el 1, pasando el flip-flop a estado SET. Cuando se elimina el 1 de la entrada, el flip-flop permanece en dicho estado SET, quedando almacenado el 1. Como se ilustra en la Figura 9.1, el procedimiento que se utiliza para almacenar un 0 es similar y pone en estado RESET al flip-flop.

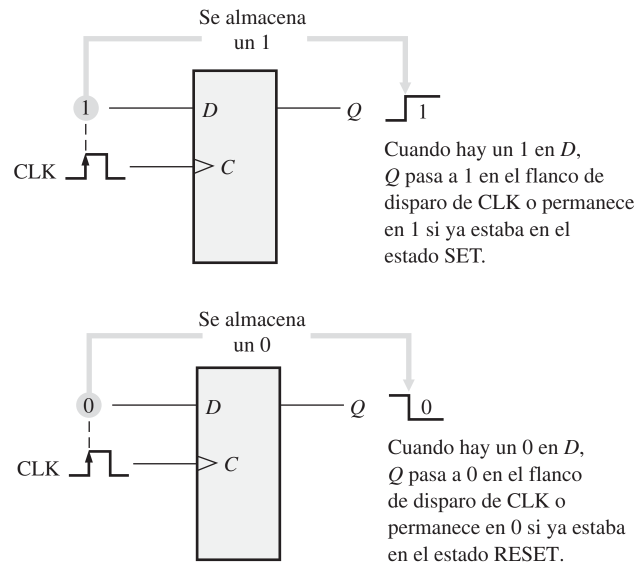


FIGURA 9.1 El flip-flop como elemento de almacenamiento.

La *capacidad de almacenamiento* de un registro es el número total de bits (1s y 0s) de un dato digital que puede contener. Cada ***etapa*** (flip-flop) de un registro de desplazamiento representa un bit de su capacidad de almacenamiento; por tanto, el número de etapas de un registro determina su capacidad de almacenamiento.

La capacidad de ***desplazamiento*** de un registro permite el movimiento de los datos de una etapa a otra dentro del registro, o la entrada o salida del mismo, en función de los impulsos de reloj que se apliquen.

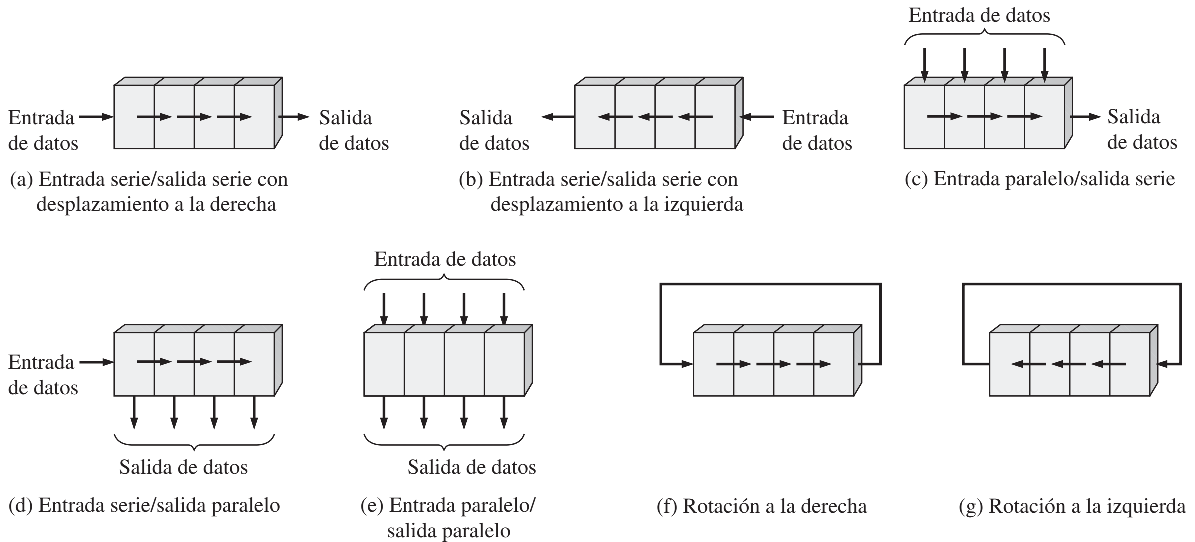


FIGURA 9.2 Movimientos básicos de datos en los registros de desplazamiento (en este ejemplo se emplean cuatro bits. Los bits se desplazan en la dirección indicada por las flechas).

La Figura 9.2 ilustra la forma en que se pueden mover los datos en los registros de desplazamiento. El bloque representa un registro cualquiera de 4 bits y las flechas indican la dirección en que se mueven los datos.

REVISIÓN DE LA SECCIÓN 9.1

Las respuestas se encuentran al final del capítulo

1. Generalmente, ¿cuál es la diferencia entre un contador y un registro de desplazamiento?
2. ¿Cuáles son las dos principales funciones que realiza un registro de desplazamiento?

9.2. REGISTROS DE DESPLAZAMIENTO CON ENTRADA Y SALIDA SERIE

Los registros de desplazamiento con entrada y salida serie aceptan datos en serie, es decir, un bit cada vez por una única línea. La información almacenada es entregada a la salida también en forma serie.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Explicar cómo se introducen en serie los bits de datos en un registro de desplazamiento. ■ Describir cómo se desplazan los bits de datos a través del registro. ■ Explicar cómo los bits de datos salen en serie del registro de desplazamiento. ■ Desarrollar y analizar los diagramas de tiempos para los registros con entrada y salida serie

En primer lugar, vamos a ver la introducción en serie de datos en un registro de desplazamiento típico. La Figura 9.3 muestra un dispositivo de 4 bits implementado con flip-flops D. Con cuatro etapas, este registro puede almacenar hasta cuatro bits de datos.

La Figura 9.4 ilustra la introducción en el registro de cuatro bits, 1010, comenzando por el bit más a la derecha. Inicialmente, el registro se borra (CLEAR). Se aplica un 0 en la línea de entrada de datos, lo que hace

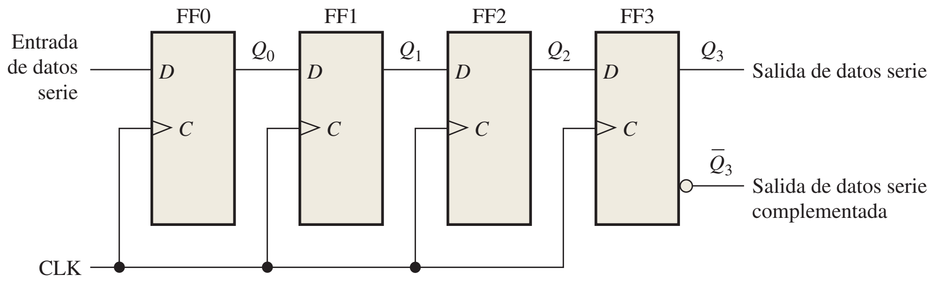


FIGURA 9.3 Registro de desplazamiento con entrada y salida serie.

$D = 0$ en el flip-flop FF0. Cuando se aplica el primer impulso de reloj, FF0 pasa al estado RESET, almacenando el 0.

A continuación se aplica a la entrada de datos el segundo bit que, en este caso, es 1, lo que hace que $D = 1$ en FF0 y $D = 0$ en FF1 debido a que la entrada D de FF1 está conectada a la salida Q_0. Cuando se produce el segundo impulso de reloj, el 1 de la entrada de datos de FF0 se desplaza, pasando este flip-flop al estado SET, y el 0 que había en FF0 se desplaza a FF1.

El tercer bit, un 0, se introduce por la línea de entrada de datos y se aplica un impulso de reloj. El 0 entra en FF0, el 1 almacenado en éste se desplaza a FF1 y el 0 almacenado en FF1 se desplaza a FF2.

El último bit, que es un 1, se aplica a la entrada de datos y se aplica el siguiente impulso de reloj. Ahora el 1 entra en FF0, el 0 almacenado en éste se desplaza a FF1, el 1 almacenado en FF1 se desplaza a FF2, y el

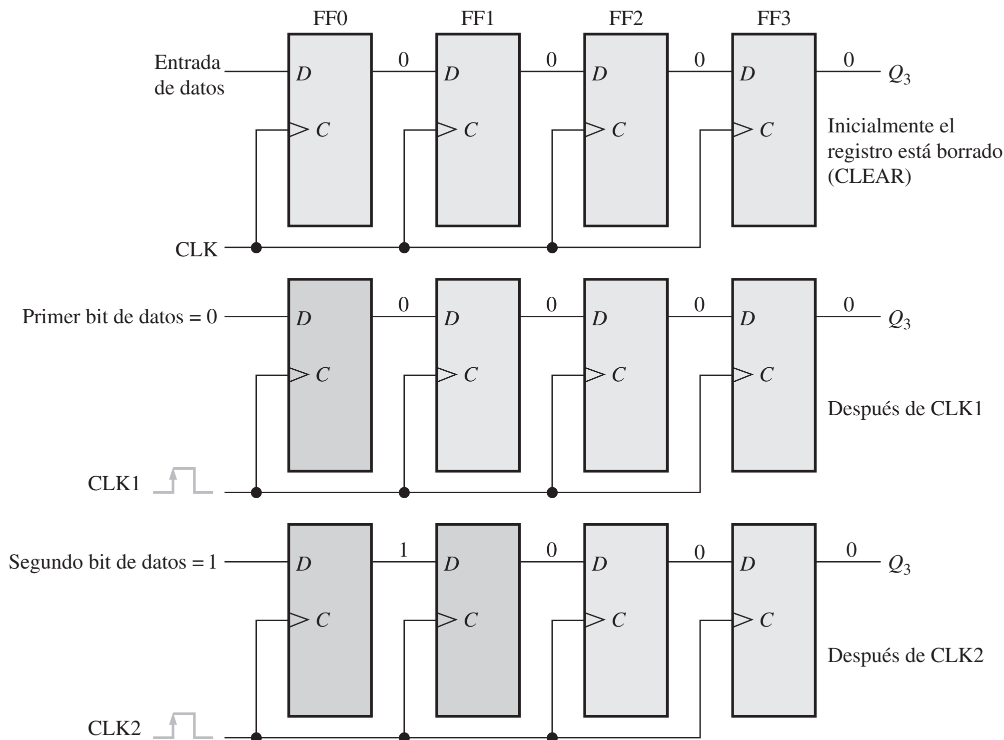


FIGURA 9.4 Introducción de cuatro bits en serie (1010) en el registro. (*Continúa*)

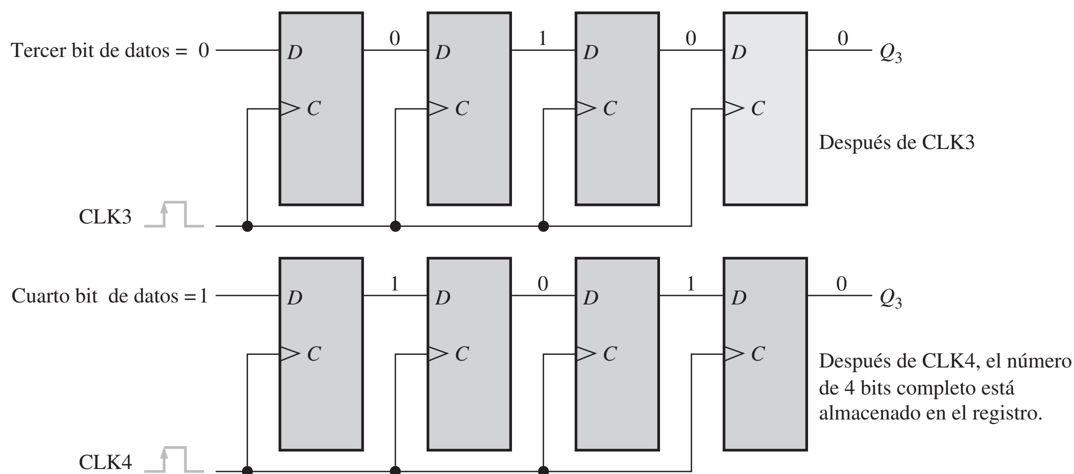


FIGURA 9.4 Introducción de cuatro bits en serie (1010) en el registro. (*Continuación*)

▲ *Con datos en serie, se transfiere un bit cada vez.*

0 almacenado en FF2 se desplaza a FF3. Esto completa la introducción en serie de los cuatro bits en el registro de desplazamiento, donde pueden quedar almacenados el tiempo que se desee, siempre que los flip-flops estén alimentados con la tensión continua necesaria.

Si se desea extraer los datos del registro, los bits deben desplazarse en serie hasta la salida Q_3, como se ilustra en la Figura 9.5. Después del cuarto impulso de reloj CLK4, el bit más a la derecha, 0, está en la salida Q_3. Si se aplica un quinto impulso de reloj, CLK5, el segundo bit aparecerá en la salida Q_3. El impulso de reloj CLK6 desplaza el tercer bit a la salida y el séptimo impulso de reloj (CLK7) desplaza el cuarto bit a la salida. Observe que, mientras que los cuatro bits iniciales se desplazan a la salida, se pueden introducir otros bits de datos. En la figura se muestra cómo se ha desplazado una serie de ceros.

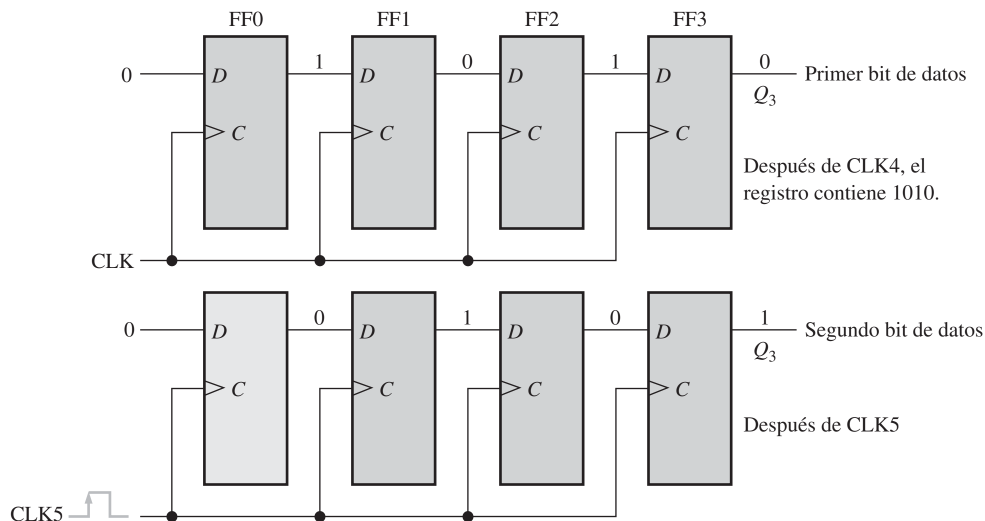


FIGURA 9.5 Los cuatro bits (1010) se han desplazado en serie a la salida del registro y se han reemplazado por ceros. (*Continúa*)

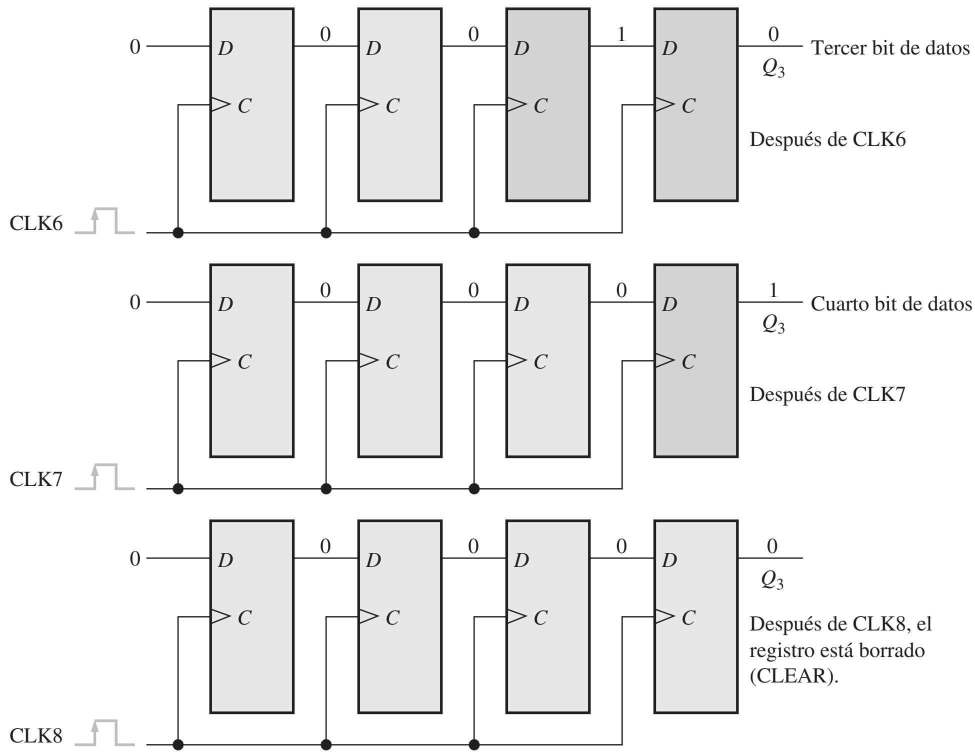


FIGURA 9.5 Los cuatro bits (1010) se han desplazado en serie a la salida del registro y se han reemplazado por ceros. (*Continuación*)



NOTAS INFORMÁTICAS

Frecuentemente, es necesario *borrar* un registro interno de una computadora. Por ejemplo, un registro puede borrarse antes de realizar una operación aritmética o de otro tipo. Un método para borrar los registros de una computadora consiste en utilizar software para extraer los contenidos del registro. Por supuesto, el resultado siempre será cero. Por ejemplo, una instrucción que realiza esta operación es SUB AL,AL. Con esta instrucción, el registro denominado AL se borrará.

EJEMPLO 9.1

Obtener los estados del registro de 5 bits de la Figura 9.6(a) para las señales de entrada de datos y la señal de reloj indicadas. Suponer que, inicialmente, el registro se ha borrado (su contenido es todo ceros).

Solución Se introduce el primer bit de datos (1) en el registro con el primer impulso de reloj y luego se desplaza de izquierda a derecha. Del mismo modo se introducen y desplazan los restantes bits. Después de cinco impulsos de reloj el registro contiene $Q_4Q_3Q_2Q_1Q_0$ = 11010. *Véase* la Figura 9.6(b).

Problema relacionado* Obtener los estados del registro si se invierte la entrada de datos. Inicialmente el registro se borra.

* Las respuestas se encuentran al final del capítulo.

FIGURA 9.6

En la Figura 9.7 se muestra el símbolo lógico tradicional de un registro de desplazamiento de 8 bits con entrada y salida serie. La designación "SRG 8" indica que es un registro de desplazamiento (SRG, *Shift Register*) con una capacidad de 8 bits.

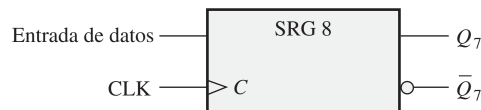


FIGURA 9.7 Símbolo lógico de un registro de desplazamiento de 8 bits con entrada y salida serie.

REVISIÓN DE LA SECCIÓN 9.2

1. Desarrollar el diagrama lógico del registro de desplazamiento de la Figura 9.3, utilizando flip-flops J-K que reemplacen a los flip-flops D.
2. ¿Cuántos impulsos de reloj se requieren para introducir un byte de datos en serie en un registro de desplazamiento de 8 bits?

9.3. REGISTROS DE DESPLAZAMIENTO CON ENTRADA SERIE Y SALIDA PARALELO

En este tipo de registro los bits de datos se introducen en serie (empezando por el bit situado más a la derecha), del mismo modo que se ha visto en la Sección 9.2. La diferencia está en la forma en que dichos bits se extraen del registro; en un registro con salida paralelo, se dispone de la salida de cada etapa. Una vez que los datos se han almacenado, cada bit se presenta en su respectiva línea de salida, estando disponibles todos los bits simultáneamente, en lugar de bit a bit como en el caso de la salida serie.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Explicar cómo se extraen los bits de datos en un registro de desplazamiento con salida paralelo. ■ Comparar la salida serie y la salida paralelo. ■ Utilizar el registro de desplazamiento de 8 bits 74HC164. ■ Desarrollar y analizar los diagramas de tiempos de los registros con entrada serie-salida paralelo

La Figura 9.8 muestra un registro de desplazamiento de 4 bits con entrada serie-salida paralelo, y su símbolo lógico.

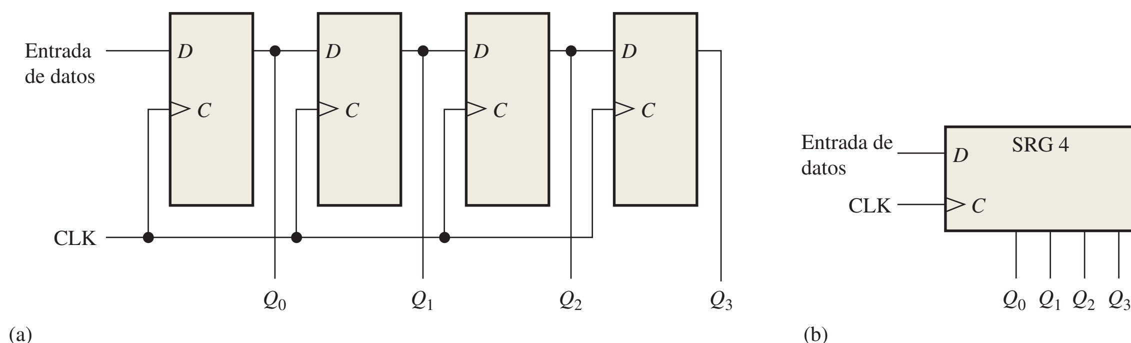


FIGURA 9.8 Registro de desplazamiento con entrada serie-salida paralelo.

EJEMPLO 9.2

Mostrar los estados del registro de 4 bits (SRG 4) para las formas de onda de entrada y de reloj de la Figura 9.9(a). Inicialmente, el contenido del registro es todo 1s.

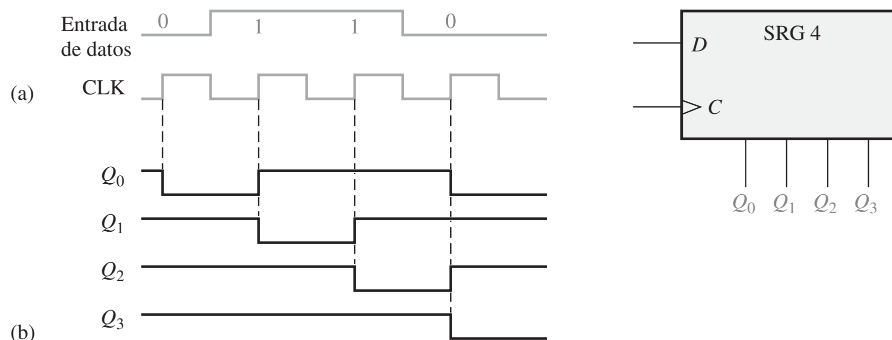


FIGURA 9.9

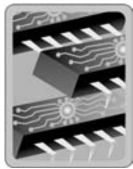
Solución Después de cuatro impulsos de reloj, el registro está en el estado 0110. *Véase* la Figura 9.9(b).

Problema relacionado Si la entrada de datos después del cuarto impulso de reloj se mantiene a 0, ¿cuál será el estado del registro después de tres impulsos de reloj más?

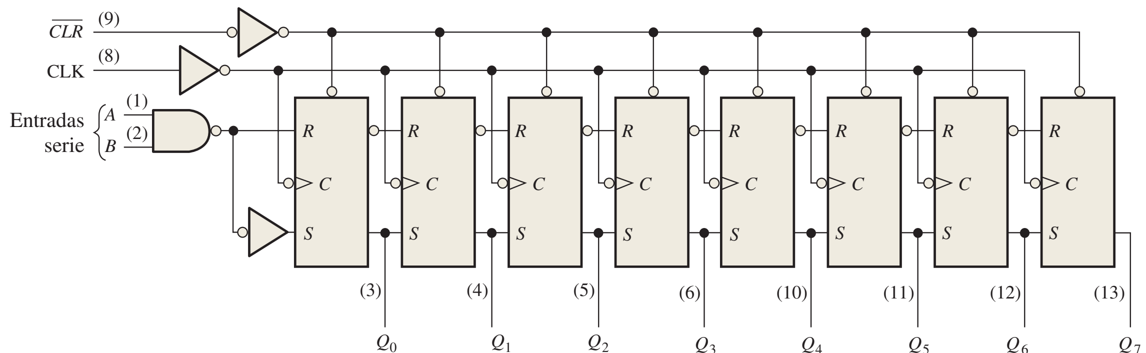
REVISIÓN DE LA SECCIÓN 9.3

1. La secuencia de bits 1101 se introduce en serie (primero el bit más a la derecha) en un registro de desplazamiento de 4 bits con salida paralelo, que inicialmente se ha borrado. ¿Cuáles son las salidas *Q* después de dos impulsos de reloj?
2. ¿Puede utilizarse un registro con entrada serie-salida paralelo como registro con entrada y salida serie?

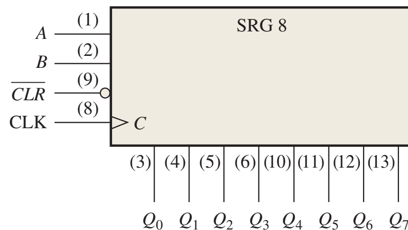
REGISTRO DE DESPLAZAMIENTO DE 8 BITS CON ENTRADA SERIE Y SALIDA PARALELO 74HC164



El 74HC164 es un ejemplo de registro de desplazamiento en formato de circuito integrado, que funciona con entrada serie-salida paralelo. En la Figura 9.10(a) se muestra su diagrama lógico y en la parte (b) el símbolo lógico típico. Observe que este dispositivo dispone de dos entradas serie, *A* y *B,* a través de una puerta, y una entrada de borrado activa a nivel BAJO ($\overline{CLR}$). Las salidas paralelo son Q_0 hasta Q_7.



(a) Diagrama lógico



(b) Símbolo lógico

FIGURA 9.10 El registro de desplazamiento de 8 bits con entrada serie-salida paralelo 74HC164.

En la Figura 9.11 se muestra un sencillo diagrama de tiempos para el 74HC164. Observe que los datos de entrada serie de la entrada *A* se desplazan al interior y a través del registro después de que la entrada *B* pasa a nivel ALTO.

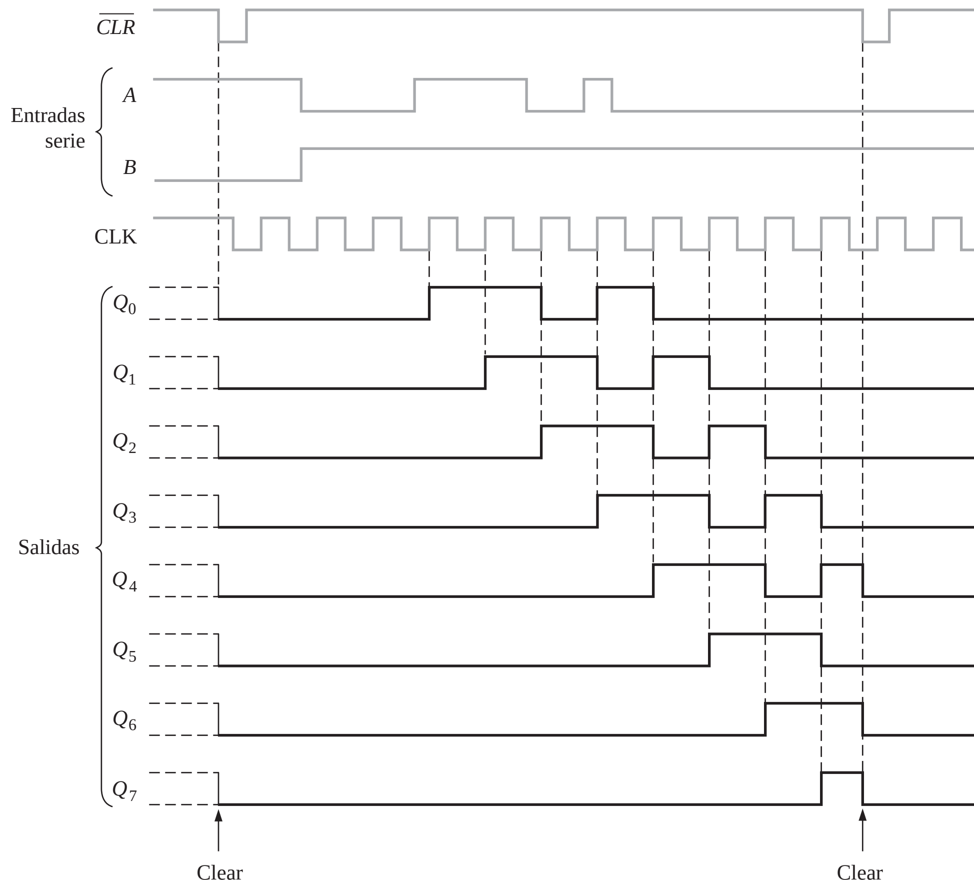


FIGURA 9.11 Diagrama de tiempos de ejemplo para un registro de desplazamiento 74HC164.

9.4 REGISTROS DE DESPLAZAMIENTO CON ENTRADA PARALELO Y SALIDA SERIE

En un registro con entradas de datos paralelo, los bits se introducen simultáneamente en sus respectivas etapas a través de líneas paralelo, en lugar de bit a bit a través una única línea como ocurre con las entradas de datos serie. La salida serie se hace del mismo modo que se ha descrito en la Sección 9.2, una vez que todos los datos están almacenados en el registro.

Al finalizar esta sección, el lector deberá ser capaz de:

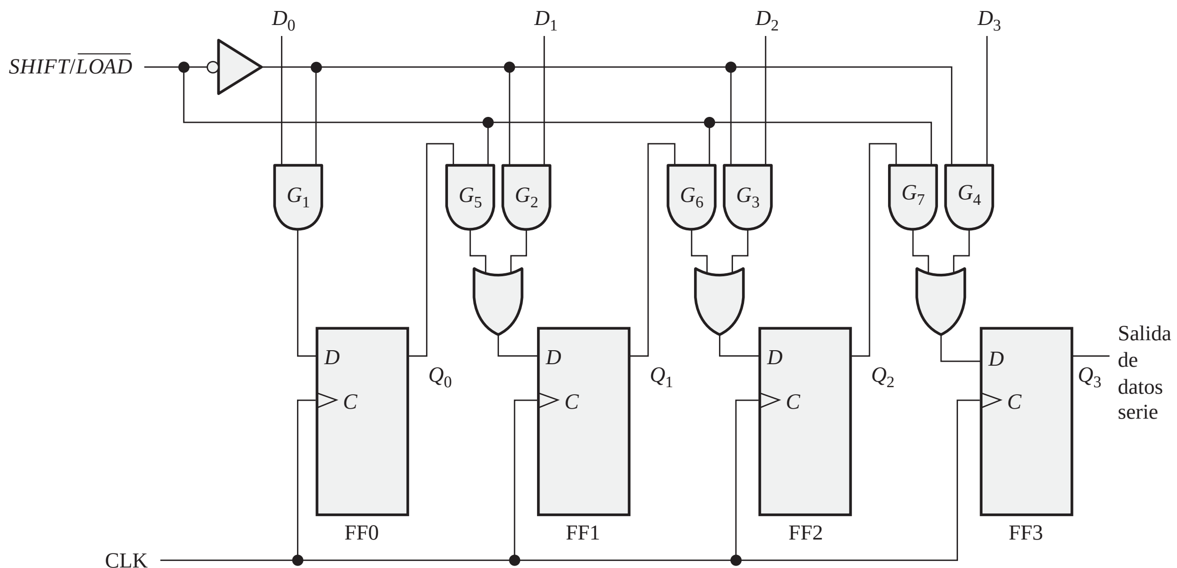
■ Explicar cómo se introducen los datos en el registro de desplazamiento con entrada paralelo. ■ Comparar las entradas serie y paralelo. ■ Utilizar el registro de desplazamiento de 8 bits de carga

paralelo 74HC165. ■ Desarrollar y analizar los diagramas de tiempos de los registros con entrada paralelo-salida serie.

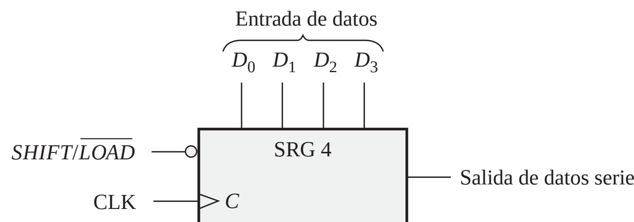
▲ *Con datos en paralelo, múltiples bits se transfieren simultáneamente.*

La Figura 9.12 ilustra un registro de desplazamiento de 4 bits con entrada paralelo-salida serie y su símbolo lógico típico. Observe que tiene cuatro líneas de entrada de datos D_0, D_1, D_2 y D_3 y una entrada $SHIFT/\overline{LOAD}$ (desplazamiento/carga), que permite ***cargar*** en paralelo los cuatro bits de datos en el registro. Cuando $SHIFT/\overline{LOAD}$ está a nivel BAJO, las puertas G_1 a G_3 se activan, permitiendo que cada bit sea aplicado a la entrada *D* de su respectivo flip-flop. Cuando se aplica un impulso de reloj, los flip-flops con $D = 1$ pasan al estado SET, y los flip-flops con $D = 0$ pasan al estado RESET, almacenándose de este modo los cuatro bits simultáneamente.

Cuando la entrada $SHIFT/\overline{LOAD}$ está a nivel ALTO, las puertas G_1 a G_4 se inhiben y las puertas G_5 a G_7 se activan, permitiendo que los bits de datos se desplacen hacia la derecha, pasando de una etapa a la siguiente. Las puertas OR permiten el desplazamiento normal o la introducción de datos en paralelo, dependiendo de qué puertas AND se hayan activado según el nivel de la entrada $SHIFT/\overline{LOAD}$. Observe que FF0 dispone de una sola puerta AND para desactivar la entrada paralelo, D_0. No precisa una implementación AND/OR ya que no hay entrada de datos en serie.



(a) Diagrama lógico



(b) Símbolo lógico

FIGURA 9.12 Registro de desplazamiento de 4 bits con entrada paralelo-salida serie.

EJEMPLO 9.3

Determinar la forma de onda de la salida de datos de un registro de 4 bits para las formas de onda de entrada paralelo de datos, de reloj y $SHIFT/\overline{LOAD}$ de la Figura 9.13(a). Utilizar el diagrama lógico de la Figura 9.12(a).

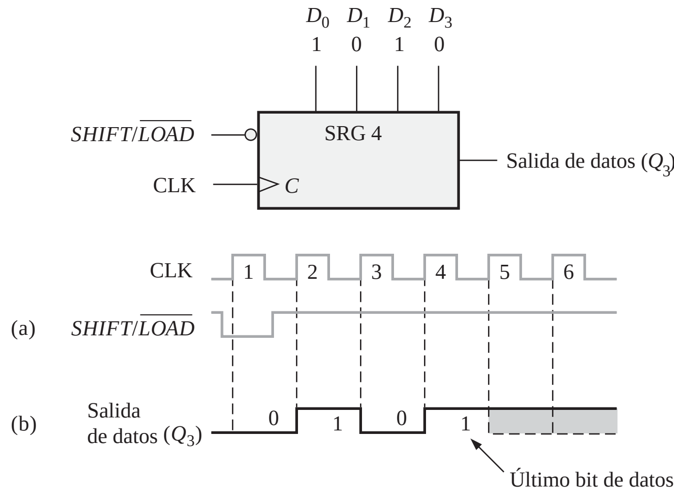


FIGURA 9.13

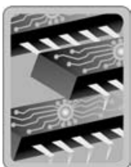
Solución En el impulso de reloj 1, los datos paralelo ($D_0D_1D_2D_3$ = 1010) se cargan en el registro, poniendo la salida Q_3 a 0. En el impulso de reloj 2, el 1 de Q_2 se desplaza a Q_3; en el impulso de reloj 3, el 0 se desplaza a Q_3; en el impulso de reloj 4, el último bit de datos (1) se desplaza a Q_3 y en el impulso de reloj 5 todos los bits se han desplazado y salido del registro, y sólo quedan 1s en el mismo (suponiendo que la entrada D_0 permanece a 1). *Véase* la Figura 9.13(b).

Problema relacionado Determinar la señal de salida de datos para las señales de reloj y $SHIFT/\overline{LOAD}$ de la Figura 9.13(a), si las entradas de datos paralelo son $D_0D_1D_2D_3 = 0101$.

REVISIÓN DE LA SECCIÓN 9.4

1. Explicar la función de la entrada $SHIFT/\overline{LOAD}$.
2. En un registro de desplazamiento 74HC165, la operación de carga paralelo ¿es síncrona o asíncrona? ¿qué quiere decir esto?

REGISTRO DE DESPLAZAMIENTO DE 8 BITS CON CARGA PARALELO 74HC165



El 74HC165 es un ejemplo de registro de desplazamiento en formato de circuito integrado que trabaja con entrada paralelo y salida serie (también puede funcionar con entrada y salida serie). La Figura 9.14 muestra el diagrama lógico interno de este dispositivo y en la parte (b) se presenta su símbolo lógico. La entrada $SHIFT/\overline{LOAD}$ ($SH/\overline{LD}$) a nivel BAJO activa todas las puertas NAND para realizar la carga paralelo. Cuando un



(a) Diagrama lógico



(b) Símbolo lógico

FIGURA 9.14 El registro de desplazamiento de 8 bits con carga paralelo 74HC165.

bit de datos de entrada es un 1, el flip-flop pasa al estado SET de forma asíncrona debido al nivel BAJO en la salida de la puerta superior. Cuando un bit de datos de entrada es un 0, el flip-flop pasa al estado RESET de forma asíncrona debido al nivel BAJO en la salida de la puerta inferior. Además, los datos se pueden introducir en serie a través de la entrada *SER*. El reloj se puede inhibir en cualquier instante aplicando un nivel ALTO a la entrada *CLK INH*. Las salidas de datos serie del registro son Q_7 y su complemento $\overline{Q}_7$. Esta implementación es distinta de la que se ha visto anteriormente, el método síncrono de carga paralelo, lo que demuestra que existen varias formas de realizar la misma función.

La Figura 9.15 es un diagrama de tiempos que muestra un ejemplo de funcionamiento de un registro de desplazamiento 74HC165.

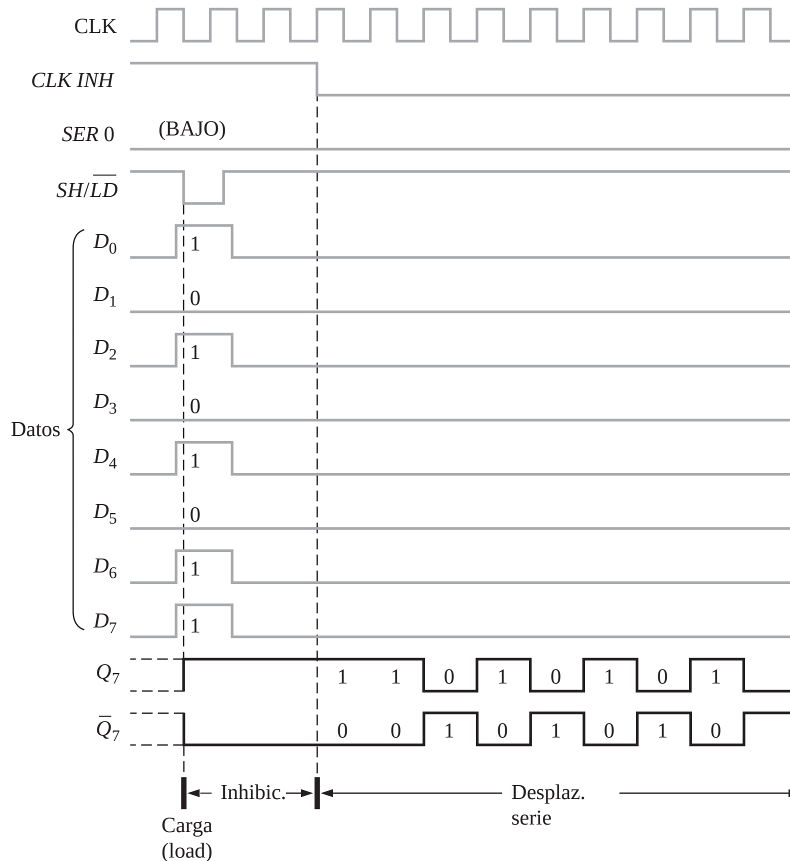


FIGURA 9.15 Diagrama de tiempos para un registro de desplazamiento 74HC165.

9.5. REGISTROS DE DESPLAZAMIENTO CON ENTRADA Y SALIDA PARALELO

En la Sección 9.4 se ha descrito la entrada en paralelo de datos y la salida en paralelo de datos también se ha visto anteriormente. El registro de entrada y salida paralelo aplica ambos métodos. Inmediatamente después de introducir simultáneamente todos los bits de datos, éstos aparecen en paralelo en las salidas paralelo.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Utilizar el registro de desplazamiento de 4 bits de acceso paralelo 74HC195. ■ Desarrollar y analizar el diagrama de tiempos para los registros de entrada y salida paralelo.

La Figura 9.16 presenta un registro de entrada y salida paralelo.

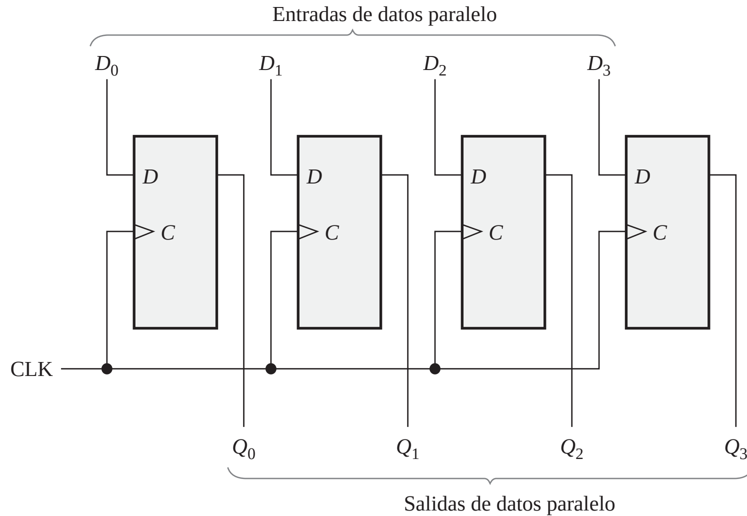
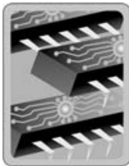


FIGURA 9.16 Registro de entrada y salida paralelo.

REGISTRO DE DESPLAZAMIENTO DE 4 BITS DE ACCESO PARALELO 74HC195



El 74HC195 puede utilizarse para trabajar con entrada y salida paralelo. Dado que también dispone de una entrada serie, se puede emplear para trabajar con entrada y salida serie, o entrada serie y salida paralelo. Puede usarse para funcionar con entrada paralelo y salida serie utilizando Q_3 como salida. En la Figura 9.17 se muestra su símbolo lógico típico.

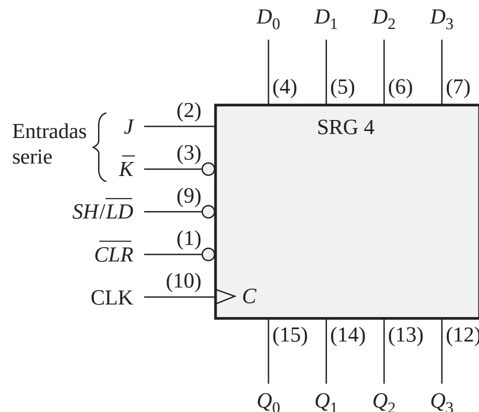


FIGURA 9.17 El registro de desplazamiento de 4 bits con acceso paralelo 74HC195.

Cuando la entrada $SHIFT/\overline{LOAD}$ ($SH/\overline{LD}$) está a nivel BAJO, los datos de las entradas paralelo se introducen en forma síncrona durante la transición positiva de la señal de reloj. Cuando $SH/\overline{LD}$ está a nivel ALTO, los datos almacenados se desplazan a la derecha (Q_0 a Q_3), sincronizados con la señal de reloj. Las entradas J y $\overline{K}$ son para las entradas de datos serie de la primera etapa del registro (Q_0); Q_3 puede utilizarse como salida de datos serie. La entrada de borrado activa a nivel BAJO es asíncrona.

El diagrama de tiempos de la Figura 9.18 ilustra el funcionamiento de este registro.

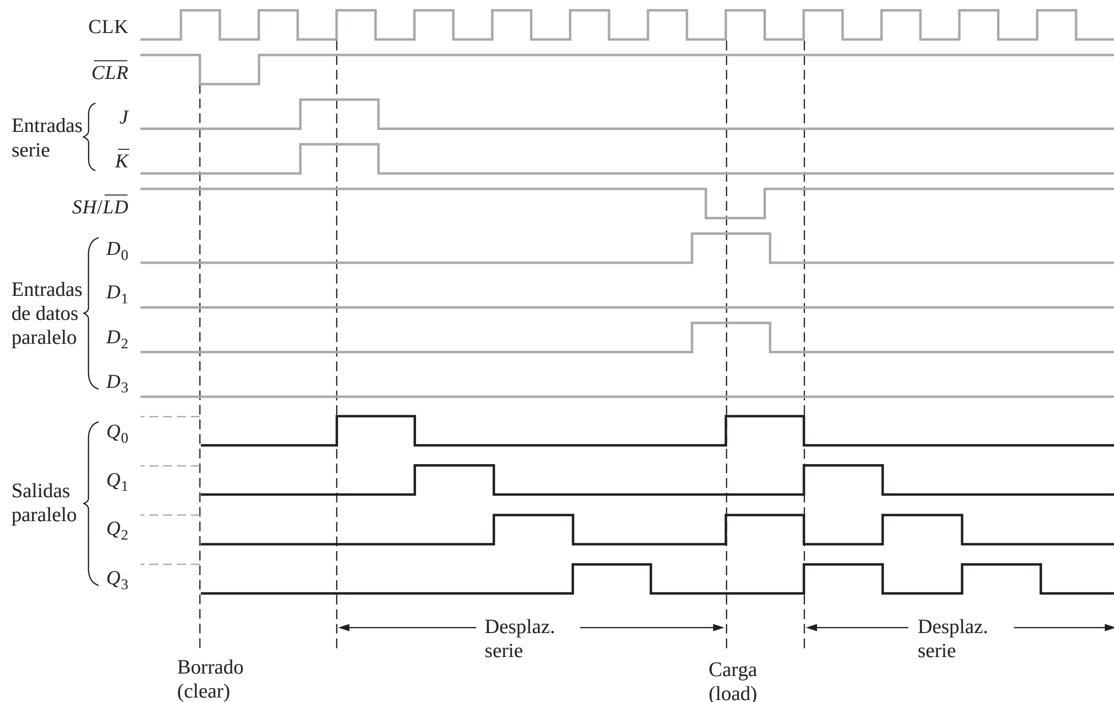


FIGURA 9.18 Diagrama de tiempos para un registro de desplazamiento 74HC195.

REVISIÓN DE LA SECCIÓN 9.5

1. En la Figura 9.16, $D_0 = 1$, $D_1 = 0$, $D_2 = 0$ y $D_3 = 1$. Después de tres impulsos de reloj, ¿cuáles son las salidas de datos?
2. Para un 74HC195, $SH/\overline{LD} = 1$, $J = 1$ y $\overline{K} = 1$. Después de un impulso de reloj, ¿cuál es el estado de Q_0?

9.6. REGISTROS DE DESPLAZAMIENTO BIDIRECCIONALES

Un registro de desplazamiento bidireccional es aquél en el que los datos se pueden desplazar a izquierda o a derecha. Se puede implementar utilizando puertas lógicas que permitan la transferencia de un bit de datos de una etapa a la siguiente de la izquierda o de la derecha, dependiendo del nivel de una línea de control.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Explicar el funcionamiento de un registro de desplazamiento bidireccional. ■ Utilizar el registro universal de desplazamiento bidireccional de 4 bits 74HC194. ■ Desarrollar y analizar los diagramas de tiempos de los registros de desplazamiento bidireccionales

En la Figura 9.19 se muestra un registro de desplazamiento ***bidireccional***. Un nivel ALTO en la entrada de control $RIGHT/\overline{LEFT}$ *(derecha/izquierda)* permite a los bits de datos que están dentro del registro des-

plazarse hacia la derecha, y un nivel BAJO hace que se desplacen hacia la izquierda. Un examen de la lógica de puertas hará evidente este funcionamiento. Cuando la entrada de control $RIGHT/\overline{LEFT}$ está a nivel ALTO, las puertas G_1 a G_4 se activan, y el estado de la salida Q de cada flip-flop pasa a la entrada D del *siguiente* flip-flop. Cuando se produce un impulso de reloj, los bits de datos se desplazan una posición a la *derecha*. Cuando esta entrada de control $RIGHT/\overline{LEFT}$ está a nivel BAJO, las puertas G_5 a G_8 se activan, y la salida Q de cada flip-flop pasa a la entrada D del flip-flop *precedente.* Cuando se genera un impulso de reloj, los bits de datos se desplazan una posición hacia la *izquierda*.

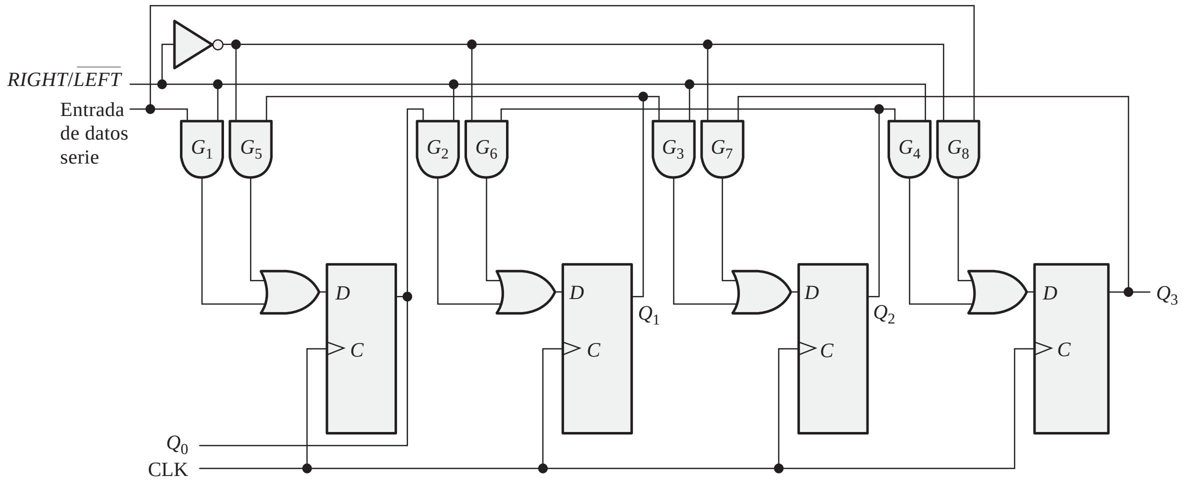


FIGURA 9.19 Registro de desplazamiento bidireccional de 4 bits.

EJEMPLO 9.4

Determinar el estado del registro de desplazamiento de la Figura 9.19 después de cada impulso de reloj para la forma de onda de la entrada de control $RIGHT/\overline{LEFT}$ indicada en la Figura 9.20(a). Suponer que $Q_0 = 1$, $Q_1 = 1$, $Q_2 = 0$ y $Q_3 = 1$, y que la línea de entrada de datos serie está a nivel BAJO.

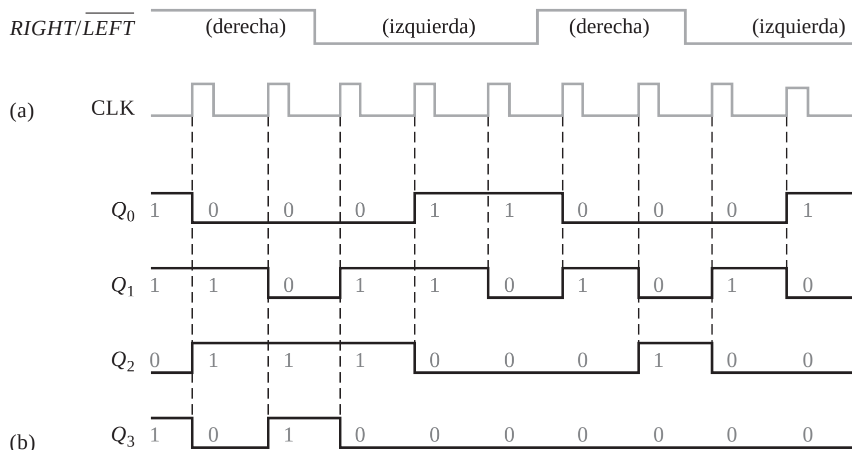


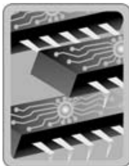
FIGURA 9.20

Solución	*Véase* la Figura 9.20(b).
Problema relacionado	Invertir la señal $RIGHT/\overline{LEFT}$ y determinar el estado del registro de desplazamiento de la Figura 9.19 después de cada impulso de reloj.

REVISIÓN DE LA SECCIÓN 9.6

1. Suponer que el registro de desplazamiento bidireccional de 4 bits de la Figura 9.19 contiene: $Q_0 = 1$, $Q_1 = 1$, $Q_2 = 0$, $Q_3 = 0$. En la línea de entrada de datos serie hay un 1. Si la entrada $RIGHT/\overline{LEFT}$ está a nivel ALTO durante tres impulsos de reloj y a nivel BAJO para otros dos impulsos más, ¿cuál será el contenido del registro después de los cinco impulsos de reloj?

REGISTRO DE DESPLAZAMIENTO UNIVERSAL BIDIRECCIONAL DE 4 BITS 74HC194



El 74HC194 es un ejemplo de un registro de desplazamiento bidireccional universal en formato de circuito integrado. Un **registro de desplazamiento universal** tiene capacidad de entrada y salida serie y paralelo. En la Figura 9.21 se muestra su símbolo lógico y en la Figura 9.22 se presenta un ejemplo de diagrama de tiempos.

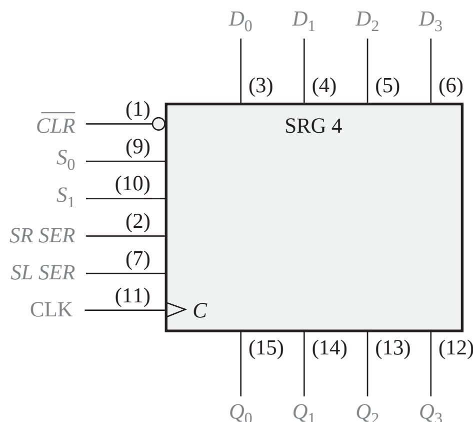


FIGURA 9.21 El registro de desplazamiento universal bidireccional de 4 bits 74HC194.

La carga paralelo, que se sincroniza con una transición positiva de la señal de reloj, se consigue aplicando los cuatro bits de datos en las entradas paralelo y un nivel ALTO en las entradas S_0 y S_1. El desplazamiento a la derecha se consigue de forma síncrona con el flanco positivo del impulso de reloj cuando S_0 está a nivel ALTO y S_1 a nivel BAJO. En este modo, los datos serie se introducen por la entrada serie de desplazamiento a la derecha (*SR SER*). Cuando S_0 está a nivel BAJO y S_1 a nivel ALTO, los bits de datos se desplazan hacia la izquierda sincronizados con la señal de reloj, introduciendo nuevos datos por la entrada serie de desplazamiento a la izquierda (*SL SER*). La entrada *SR SER* entra en la etapa Q_0 y *SL SER* entra en la etapa Q_3.

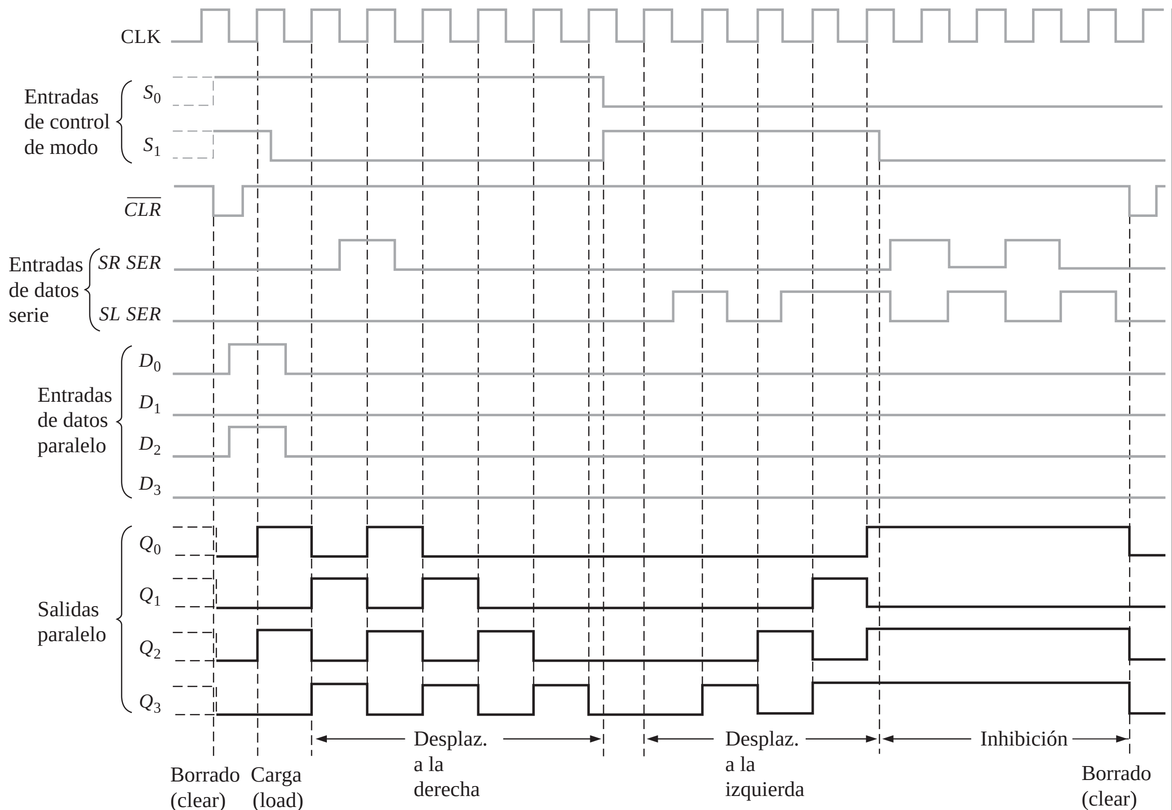


FIGURA 9.22 Diagrama de tiempos de un registro de desplazamiento 74HC194.

9.7. CONTADORES BASADOS EN REGISTRO DE DESPLAZAMIENTO

Un contador basado en un registro de desplazamiento es básicamente un registro de desplazamiento con la salida serie realimentada a la entrada serie, de modo que se generen secuencias especiales. A menudo estos dispositivos se clasifican como contadores porque disponen de una secuencia de estados específica. En esta sección, se presentan dos de los tipos más comunes de esta clase de contadores, el contador Johnson y el contador en anillo.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Establecer en qué se diferencia un contador basado en un registro de desplazamiento de un registro de desplazamiento básico. ■ Explicar el funcionamiento de un contador Johnson. ■ Especificar una secuencia de Johnson para cualquier número de bits. ■ Explicar el funcionamiento de un contador en anillo y determinar la secuencia de cualquier contador en anillo específico.

El contador Johnson

En un **contador Johnson**, el complemento de la salida del último flip-flop se conecta a la entrada *D* del primer flip-flop (también se puede implementar con otros tipos de flip-flop). Esta realimentación permite generar una secuencia de estados característica, tal y como muestran las Tablas 9.1 y 9.2 para un dispositivo de 4 bits y otro de 5 bits, respectivamente. Observe que la secuencia de 4 bits tiene un total de ocho estados, o patrones de bits, y que la secuencia de 5 bits establece un total de diez estados. En general, un contador Johnson generará un módulo de $2n$, donde n es el número de etapas del contador.

Impulso de reloj	Q_0	Q_1	Q_2	Q_3
0	0	0	0	0
1	1	0	0	0
2	1	1	0	0
3	1	1	1	0
4	1	1	1	1
5	0	1	1	1
6	0	0	1	1
7	0	0	0	1

TABLA 9.1 Secuencia Johnson de cuatro bits.

Impulso de reloj	Q_0	Q_1	Q_2	Q_3	Q_4
0	0	0	0	0	0
1	1	0	0	0	0
2	1	1	0	0	0
3	1	1	1	0	0
4	1	1	1	1	0
5	1	1	1	1	1
6	0	1	1	1	1
7	0	0	1	1	1
8	0	0	0	1	1
9	0	0	0	0	1

TABLA 9.2 Secuencia Johnson de cinco bits.

En la Figura 9.23 se muestra la implementación de los contadores Johnson de 4 y 5 etapas. La implementación de un contador Johnson es muy sencilla e independiente del número de etapas. La salida *Q* de cada etapa se conecta a la entrada *D* de la etapa siguiente (suponiendo que se utilizan flip-flops D). La única excepción es que la salida $\bar{Q}$ de la última etapa se conecta a la entrada *D* de la primera etapa. Como indican las secuencias de las Tablas 9.1 y 9.2, el contador se "llenará" de 1s de izquierda a derecha, y luego se "llenará" de nuevo de 0s.

En las Figuras 9.24 y 9.25 se muestran, respectivamente, los diagramas de tiempos de los contadores de 4 y 5 bits.

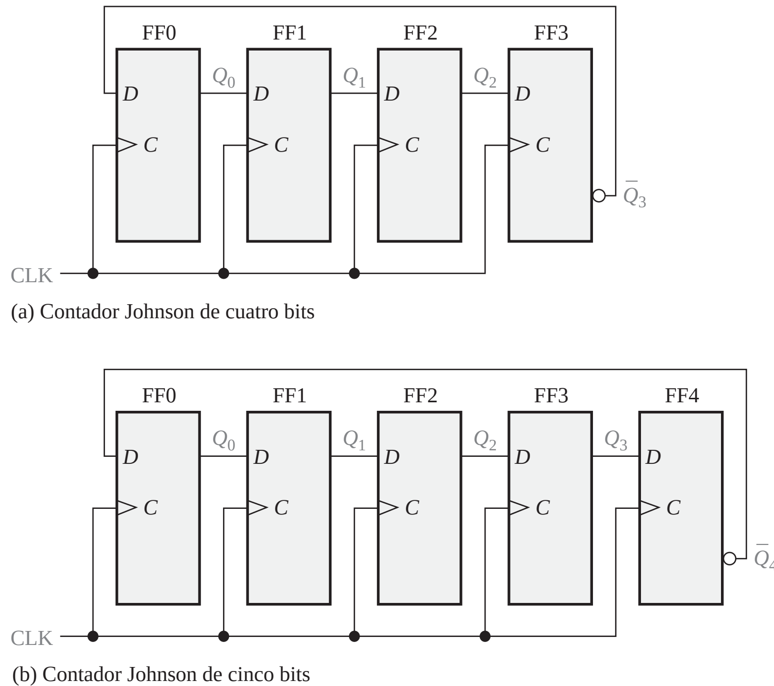


FIGURA 9.23 Contadores Johnson de cuatro y cinco bits.

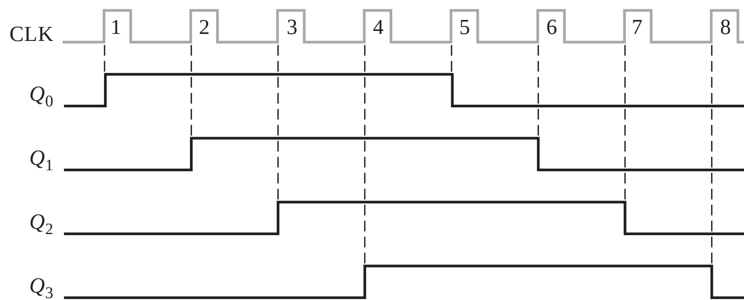


FIGURA 9.24 Secuencia de temporización del contador Johnson de 4 bits.

El contador en anillo

El **contador en anillo** utiliza un flip-flop para cada estado de su secuencia. Tiene la ventaja de que no se requieren puertas de decodificación. En el caso de un contador en anillo de 10 bits, hay una única salida para cada dígito decimal.

En la Figura 9.26 se muestra un diagrama lógico para un contador en anillo de 10 bits. En la Tabla 9.3 se facilita la secuencia de este mismo contador. Inicialmente, se presenta un 1 en el primer flip-flop, y se borran los restantes flip-flops. Observe que las conexiones entre etapas son iguales a las del contador Johnson, excepto que en este caso es la salida Q de la última etapa, en lugar de $\overline{Q}$, la que se realimenta. Las diez salidas del contador indican directamente el valor decimal de la cuenta de los impulsos de reloj. Por ejemplo, un 1 en Q_0 representa un cero, un 1 en Q_1 indica uno, un 1 en Q_2 corresponde a dos en decimal, un 1 en Q_3 corresponde

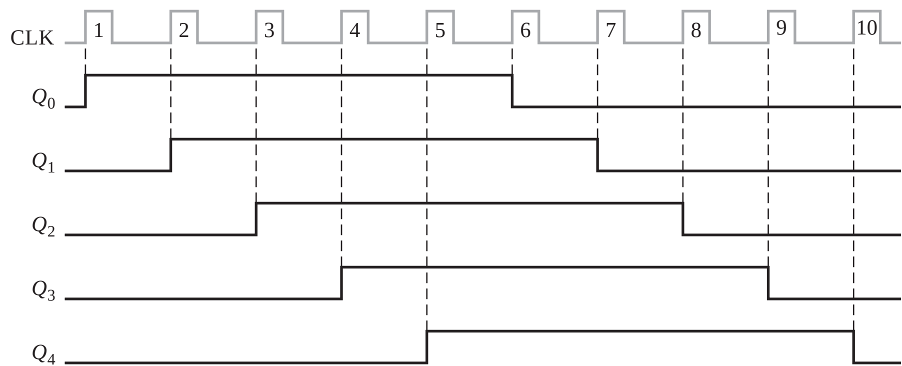


FIGURA 9.25 Secuencia de temporización del contador Johnson de 5 bits.

a tres, y así sucesivamente. Debería verificar usted mismo que sólo un 1 se mantiene en el contador y que éste simplemente se desplaza “alrededor del anillo”, avanzando una etapa con cada impulso de reloj.

Como se ilustra en el Ejemplo 9.5, se pueden conseguir otras secuencias introduciendo más de un 1 en el contador.

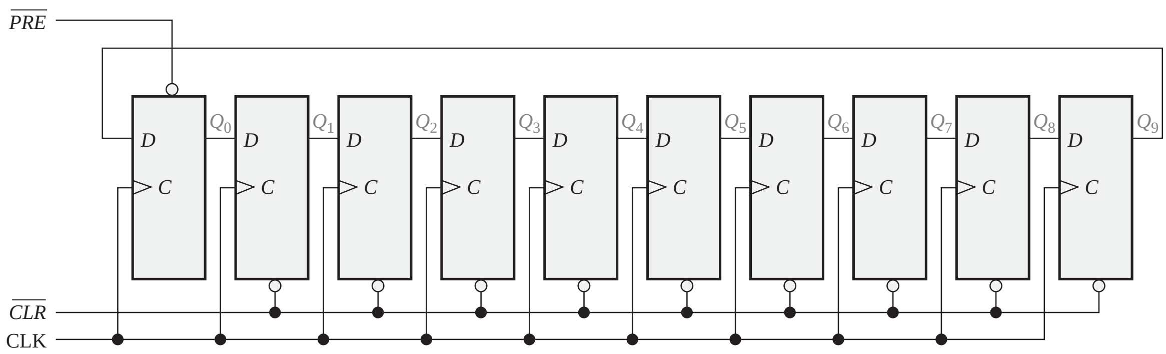


FIGURA 9.26 Contador en anillo de diez bits.

Impulso de reloj	Q_0	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7	Q_8	Q_9
0	1	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0
2	0	0	1	0	0	0	0	0	0	0
3	0	0	0	1	0	0	0	0	0	0
4	0	0	0	0	1	0	0	0	0	0
5	0	0	0	0	0	1	0	0	0	0
6	0	0	0	0	0	0	1	0	0	0
7	0	0	0	0	0	0	0	1	0	0
8	0	0	0	0	0	0	0	0	1	0
9	0	0	0	0	0	0	0	0	0	1

TABLA 9.3 Secuencia del contador en anillo de diez bits.

EJEMPLO 9.5

Si el contador en anillo de 10 bits de la Figura 9.26 tiene el estado inicial 10100000000, determinar la forma de onda para cada una de las salidas Q.

Solución *Véase* la Figura 9.27.

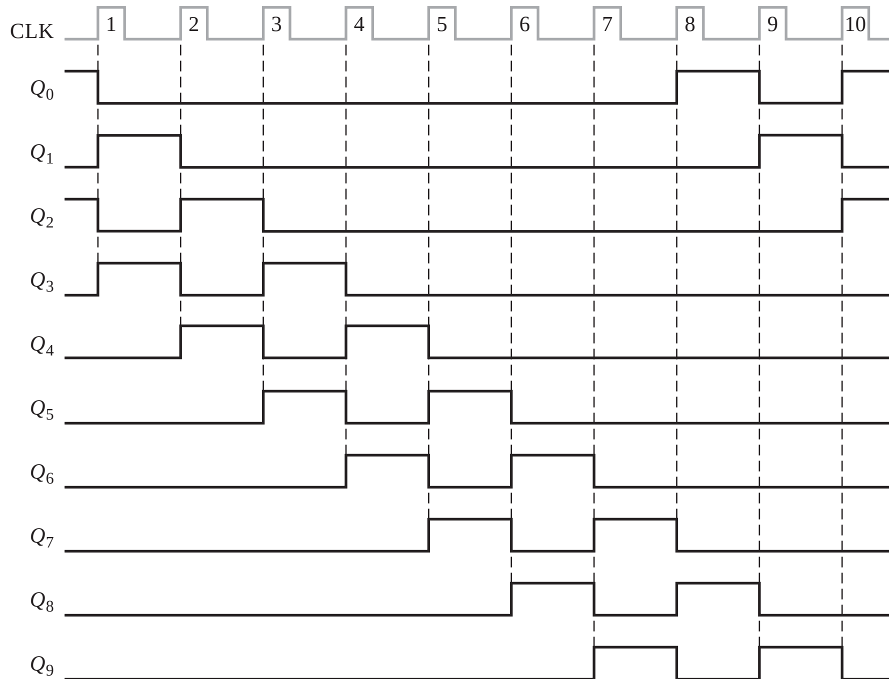


FIGURA 9.27

Problema relacionado Si un contador en anillo de 10 bits está en el estado inicial 0101001111, determinar la forma de onda de cada salida Q.

REVISIÓN DE LA SECCIÓN 9.7

1. ¿Cuántos estados tiene un contador Johnson de 8 bits?
2. Escribir la secuencia de estados de un contador Johnson de 3 bits que se inicia en el estado 000.

9.8. APLICACIONES DE LOS REGISTROS DE DESPLAZAMIENTO

Los registros de desplazamiento se encuentran en muchos tipos de aplicaciones; en esta sección, se van a ver algunas de ellas.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Utilizar un registro de desplazamiento para generar un retardo de tiempo. ■ Implementar una secuencia específica de un contador en anillo utilizando un registro de desplazamiento 74HC195. ■ Explicar cómo se usan los registros de desplazamiento para la conversión de datos serie-paralelo. ■ Definir *UART*. ■ Explicar el funcionamiento de un codificador de teclado y cómo se utilizan los registros en esta aplicación.



NOTAS INFORMÁTICAS

Los registros de propósito general en el Pentium son registros de 32 bits que se pueden utilizar para el almacenamiento temporal de datos, así como para usos específicos. Cuatro de estos registros son los siguientes: el *acumulador* (EAX), que se utiliza principalmente para el almacenamiento temporal de datos y de operandos de instrucciones; el *registro base* (EBX), que se utiliza para almacenar un valor de forma temporal; el *registro contador* (ECX), que se usa principalmente para determinar el número de repeticiones en bucles, operaciones con cadenas, desplazamientos o rotaciones. El *registro de datos* (EDX), que normalmente se emplea para el almacenamiento temporal de datos.

Retardo de tiempo

Los registros de desplazamiento con entrada y salida serie se usan para obtener un retardo de tiempo de la entrada a la salida, que es función del número de etapas (*n*) del registro y de la frecuencia de reloj.

Cuando se aplica un impulso de datos a la entrada serie de la Figura 9.28 (*A* y *B* se conectan juntas), éste se introduce en la primera etapa sincronizado con el flanco de disparo del impulso de reloj. El dato se desplaza de etapa en etapa con cada impulso de reloj sucesivo hasta que aparece en la salida serie *n* períodos de reloj más tarde. En la Figura 9.28 se ilustra este funcionamiento, utilizando un registro de desplazamiento con entrada y salida serie de 8 bits y una frecuencia de reloj de 1 MHz, para conseguir un retardo de tiempo (t_d) de 8 μs (8 × 1 μs). Este retardo se puede aumentar o disminuir variando la frecuencia de reloj. El retardo de tiempo también se puede incrementar conectando en cascada registros de desplazamiento, y se puede decrementar tomando sucesivamente la salida de las etapas intermedias del registro, si están disponibles, como se ilustra en el Ejemplo 9.6.

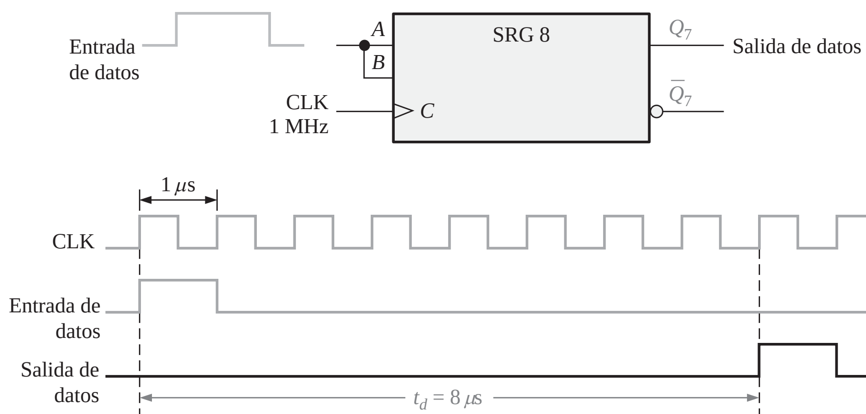


FIGURA 9.28 El registro de desplazamiento como dispositivo de retardo de tiempo.

EJEMPLO 9.6

Determinar el retardo de tiempo entre la entrada serie y cada salida del registro de la Figura 9.29. Realizar un diagrama de tiempos para ilustrarlo.

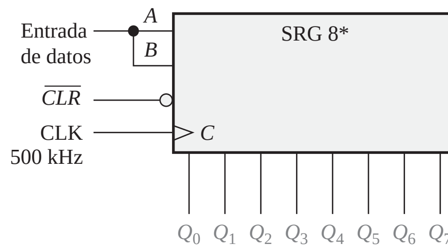


FIGURA 9.29

Solución El período de reloj es 2 μs. Luego el retardo de tiempo puede incrementarse o decrementarse de dos en dos μs, desde un mínimo de 2 μs hasta un máximo de 16 μs, como ilustra la Figura 9.30.

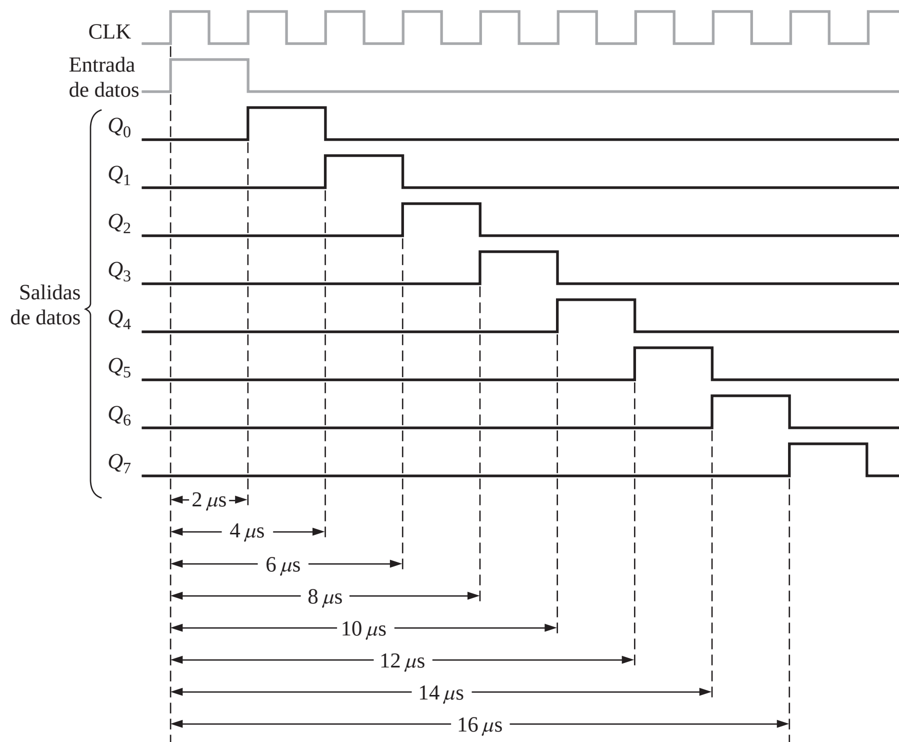


FIGURA 9.30 Diagrama de tiempos que muestra los retardos para el registro de la Figura 9.29.

Problema relacionado Determinar la frecuencia de reloj requerida para obtener un retardo de tiempo de 24 μs en la salida Q_7 del registro de la Figura 9.29.

CONTADOR EN ANILLO QUE UTILIZA UN REGISTRO DE DESPLAZAMIENTO 74HC195



Un registro de desplazamiento se puede utilizar como contador en anillo, si la salida se realimenta a la entrada serie. La Figura 9.31 ilustra esta aplicación empleando un registro de desplazamiento de 4 bits 74HC195.

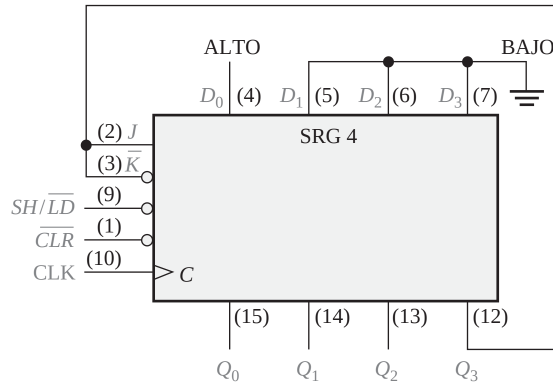


FIGURA 9.31 Un 74HC195 conectado como contador en anillo.

Inicialmente, la secuencia patrón de bits 1000 (o cualquier otra) se introduce en el contador en modo síncrono para inicializarlo, aplicando dicha secuencia patrón a las entradas de datos paralelo, con la entrada $SH/\overline{LD}$ a nivel BAJO y aplicando un impulso de reloj. Después de esta inicialización, el 1 se desplaza a través del contador en anillo, tal y como muestra el diagrama de tiempos de la Figura 9.32.

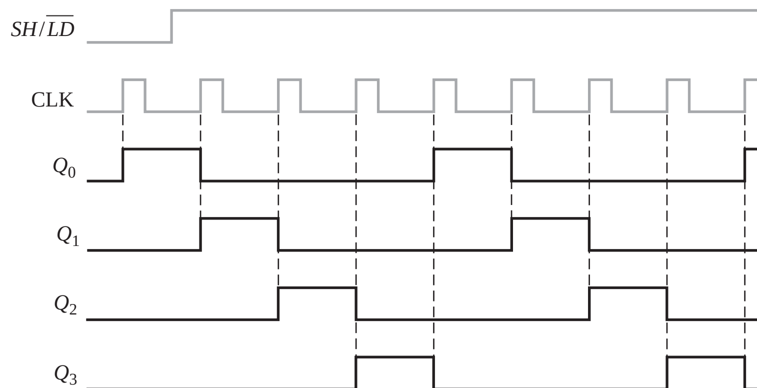


FIGURA 9.32 Diagrama de tiempos que muestra dos ciclos completos del contador en anillo de la Figura 9.31, cuando se inicializa con el estado 1000.

Convertidor de datos serie-paralelo

La transmisión de datos serie de un sistema digital a otro se usa comúnmente para reducir el número de conductores de la línea de transmisión. Por ejemplo, se pueden enviar en serie ocho bits por un único conductor, los cuales precisarían ocho conductores para transmitirse en paralelo.

Una computadora o un sistema basado en microprocesador, normalmente, requiere que la entrada de datos se haga en paralelo, por lo que es preciso realizar una conversión serie-paralelo. En la Figura 9.33 se muestra un convertidor de datos serie-paralelo simplificado, en el que se emplean dos tipos de registros de desplazamiento.

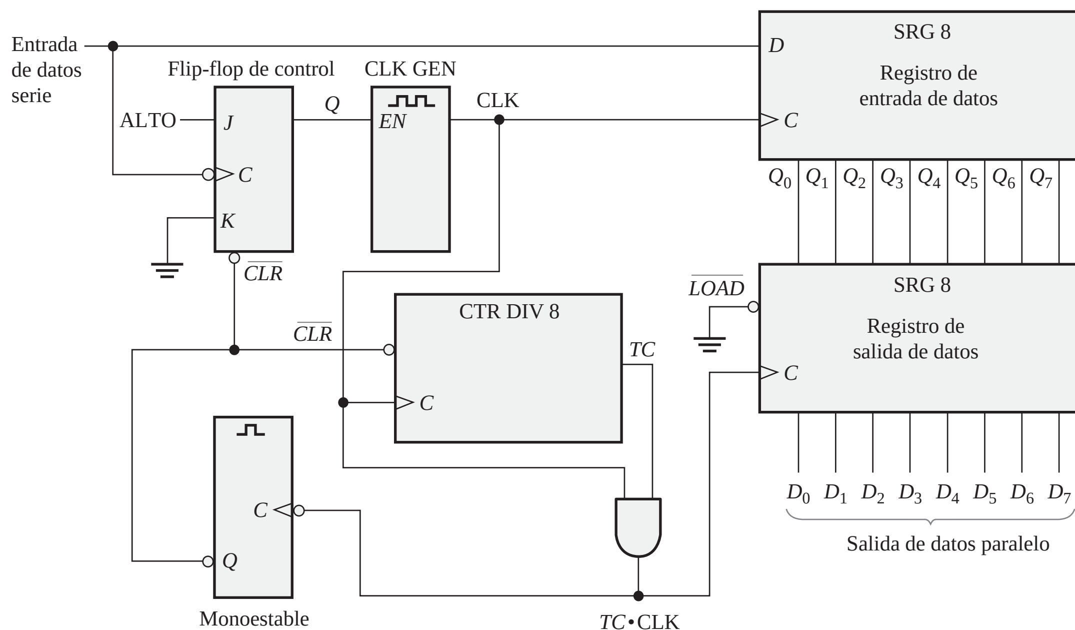


FIGURA 9.33 Diagrama lógico simplificado de un convertidor serie-paralelo.

Para ilustrar el funcionamiento del convertidor serie-paralelo, utilizaremos el formato de datos serie de la Figura 9.34, compuesto de once bits. El primer bit (bit de arranque) siempre es 0 y siempre se inicia en una transición de nivel ALTO a nivel BAJO. Los siguientes ocho bits (D_7 a D_0) son los bits de datos (uno de los bits puede ser de paridad), y los dos últimos bits (bits de parada) son siempre 1. Cuando no se transmiten datos, la línea de datos serie siempre está a 1.

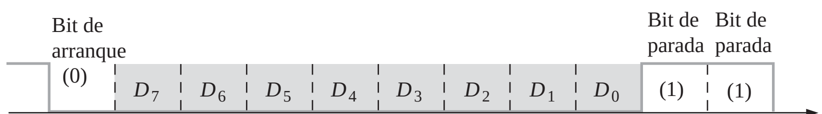


FIGURA 9.34 Formato de datos serie.

La transición de nivel ALTO a nivel BAJO del bit de arranque activa el flip-flop de control, el cual a su vez activa el generador de reloj (CLK GEN). Después de un retardo fijo, el generador de reloj comienza a generar un tren de impulsos, que se aplica al registro de entrada de datos y al contador de división por 8. La señal de reloj tiene exactamente una frecuencia igual a la de los datos serie de entrada, y el primer impulso de reloj se produce después del bit de arranque y simultáneamente con el primer bit de datos.

El diagrama de tiempos de la Figura 9.35 ilustra la siguiente operación básica: los ocho bits de datos (D_7 a D_0) se desplazan en serie a través del registro de entrada de datos. Después del octavo impulso de reloj se produce una transición de nivel ALTO a nivel BAJO en la salida *TC* del contador a la que se aplica la opera-

ción AND con la señal de reloj (*TC·CLK),* lo que hace que los ocho bits se carguen en el registro de salida de datos. Esta misma transición también dispara el monoestable, el cual produce un impulso de corta duración que borra el contador, pone en estado de RESET el flip-flop de control y desactiva el generador de reloj. Ahora, el sistema está preparado para recibir el siguiente grupo de once bits, y queda a la espera de que se produzca la siguiente transición de nivel ALTO a nivel BAJO del bit de arranque.

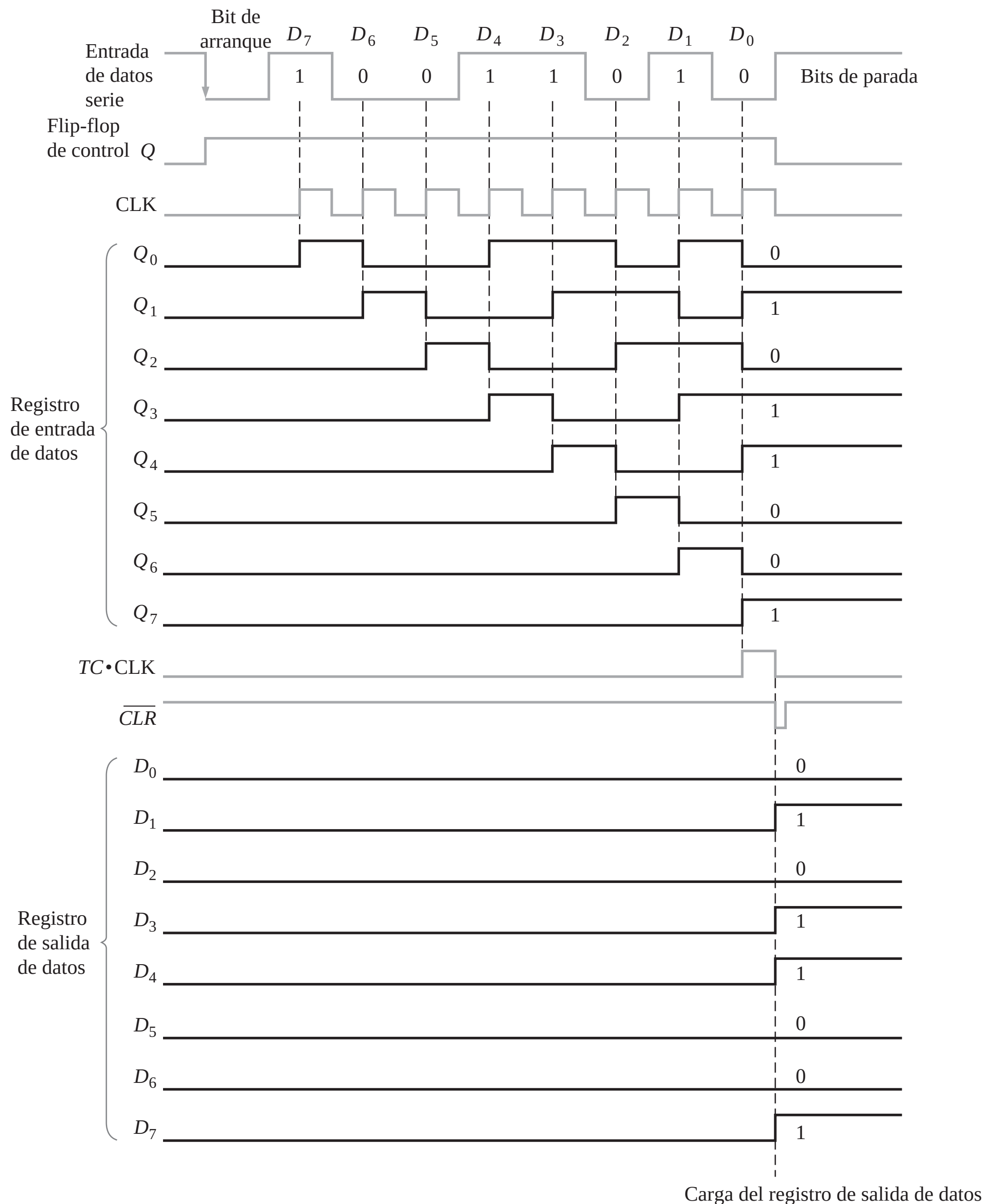


FIGURA 9.35 Diagrama de tiempos que ilustra el funcionamiento del convertidor de datos serie-paralelo de la Figura 9.33.

La conversión de datos paralelo-serie se puede realizar invirtiendo el proceso que se acaba de establecer. Sin embargo, puesto que se deben generar los datos en serie, es preciso considerar otros requisitos adicionales.

Transmisor receptor asíncrono universal (UART, Universal Asynchronous Receiver Transmitter)

Como ya se ha mencionado, las computadoras y sistemas basados en microprocesador, a menudo, transmiten y reciben datos en paralelo. Frecuentemente, estos sistemas deben comunicarse con dispositivos externos que envían y/o reciben los datos en serie. Un dispositivo que realiza la interfaz de conversión es el transmisor-receptor asíncrono universal (UART). En la Figura 9.36 se ilustra una UART en una aplicación general de un sistema basado en microprocesador.

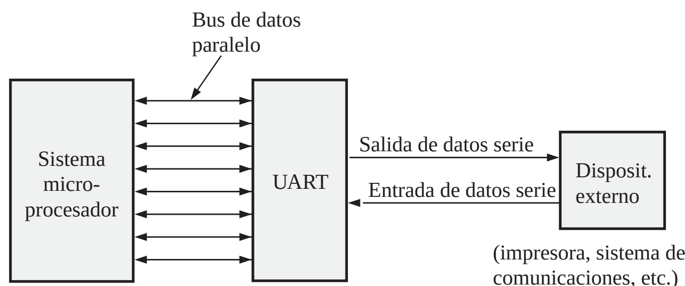


FIGURA 9.36 Interfaz de una UART.

Una UART incluye un convertidor de datos serie-paralelo, como el que hemos visto, y un convertidor de datos paralelo-serie, como muestra la Figura 9.37. Básicamente, el bus de datos es un conjunto de conductores paralelo a lo largo de los cuales se mueven los datos entre la UART y el sistema microprocesador. Los buffers establecen la interfaz entre los registros de datos y el bus de datos.

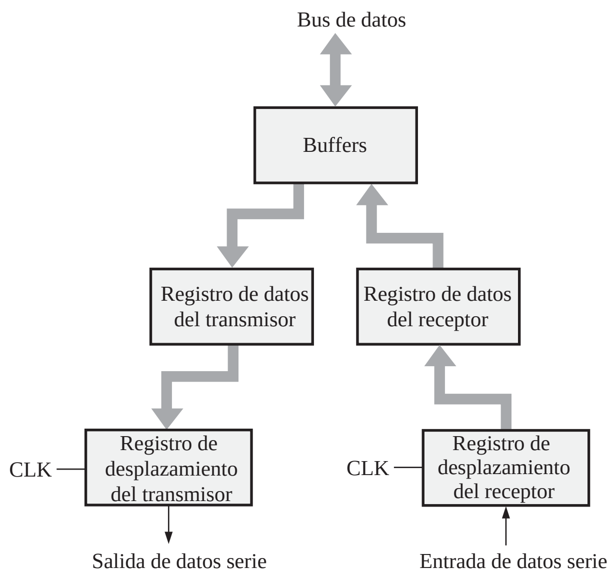


FIGURA 9.37 Diagrama de bloques básico de una UART.

La UART recibe los datos en serie, los convierte a formato paralelo y los coloca en el bus de datos. La UART también acepta datos paralelo del bus de datos, los convierte a formato serie y los transmite al dispositivo externo.

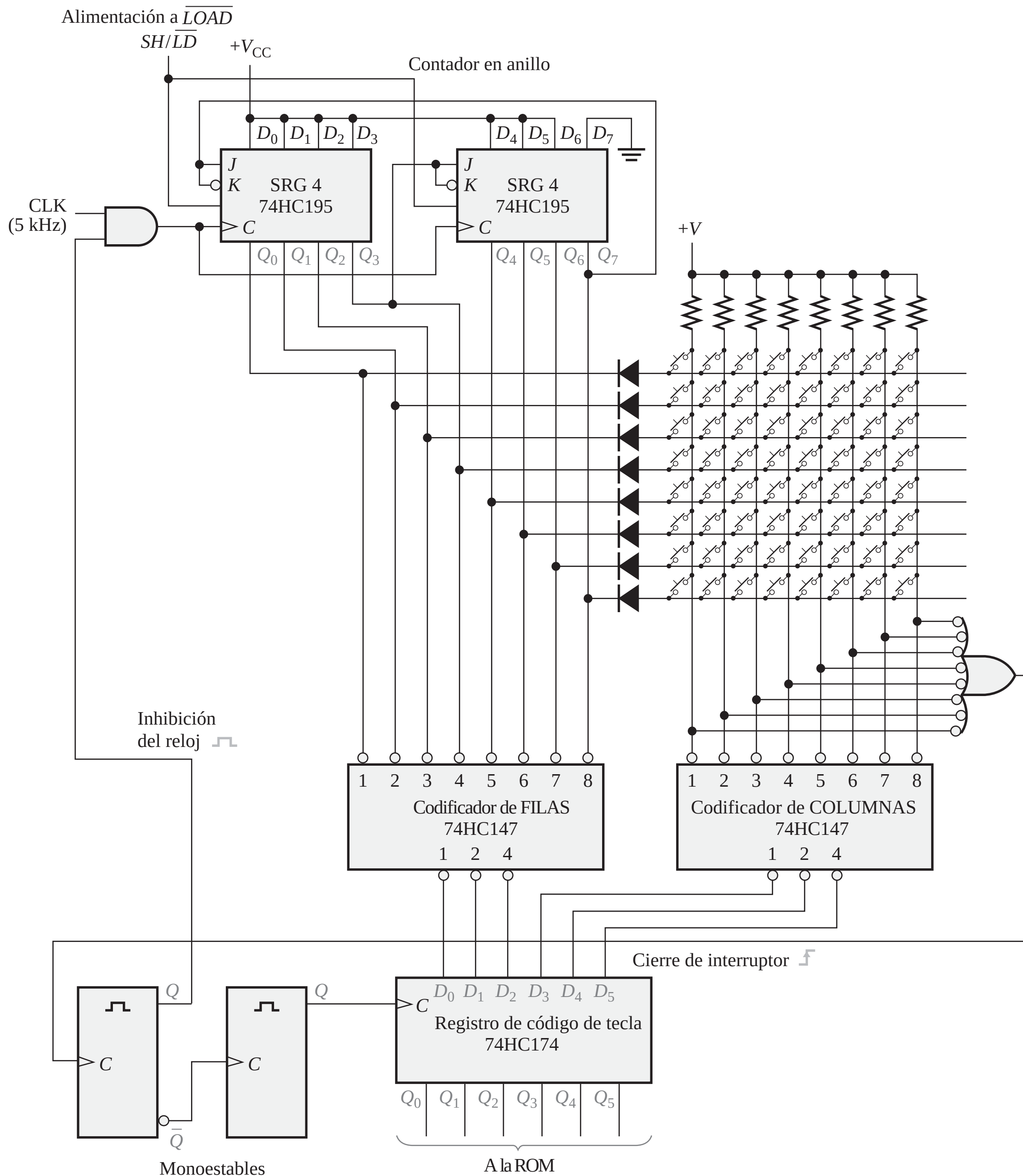


FIGURA 9.38 Circuito simplificado de codificación de teclado.

Codificador de teclado

El codificador de teclado es un buen ejemplo de aplicación de un registro de desplazamiento utilizado como contador en anillo, junto con otros dispositivos. Recuerde que, en el Capítulo 6, se presentó un codificador simplificado de teclado de computadora sin almacenamiento de datos.

La Figura 9.38 muestra un codificador de teclado simplificado que codifica la pulsación de una tecla en una matriz organizada en ocho filas y ocho columnas, que define 64 teclas. Se conectan dos registros de desplazamiento de 4 bits 74HC195 como contador en anillo de 8 bits con una secuencia patrón de bits formada por siete unos y un cero, que se activa cuando se conecta la alimentación. Se usan dos codificadores con prioridad 74HC147 (vistos en el Capítulo 6) como codificadores de ocho-líneas a tres-líneas (la entrada 9 a nivel ALTO, la salida 8 no se utiliza), para codificar las FILAS y las COLUMNAS de la matriz de teclado. El 74HC174A (séxtuple flip-flop) se usa como registro con entrada y salida paralelo en el que se almacena el código FILA/COLUMNA procedente de los codificadores con prioridad.

El funcionamiento básico del codificador de teclado de la Figura 9.38 es el siguiente: el contador en anillo "explora" las filas para detectar la pulsación de una tecla, a medida que la señal de reloj desplaza el 0 por el contador a una frecuencia de 5 kHz. Secuencialmente se aplica un 0 (nivel BAJO) a una de las líneas de FILA, mientras que las restantes líneas de FILA están a nivel ALTO. Todas las líneas FILA se conectan a las entradas del codificador de filas, de modo que la salida de 3 bits del mismo, en cualquier instante, representa, en binario, la línea FILA que está a nivel BAJO. Cuando se pulsa una tecla, la línea de COLUMNA correspondiente se conecta a la línea de FILA. Cuando el contador en anillo toma la línea de FILA que está a nivel BAJO, dicha COLUMNA también quedará a nivel BAJO. El codificador de columnas genera la salida binaria que corresponde a la COLUMNA en la que está la tecla pulsada. El código de tres bits de la FILA más el código de tres bits de la COLUMNA identifican unívocamente la tecla que se ha presionado. Este código de seis bits se aplica a las entradas del registro de código de tecla. Cuando se ha pulsado una tecla, los dos monoestables producen un impulso de reloj retrasado, para realizar la carga paralelo del código de seis bits en el registro de código de tecla. Este retraso permite que se extingan los rebotes de los contactos. La salida del primer monoestable también inhibe al contador en anillo, para evitar la exploración mientras que se están cargando los datos en el registro de códigos de las teclas.

Este código de 6 bits contenido en el registro de código de tecla se aplica ahora a una memoria ROM (*Read-Only Memory*, memoria de sólo lectura) para convertirse en un código alfanumérico apropiado que identifique los caracteres del teclado. Las memorias ROM se estudian en el Capítulo 10.

REVISIÓN DE LA SECCIÓN 9.8

1. En el codificador de teclado, ¿cuántas veces por segundo explora el contador en anillo el teclado?
2. ¿Cuál es el código de 6 bits FILA/COLUMNA (código de tecla) para la fila superior y la columna más a la izquierda del codificador de teclado?
3. ¿Cuál es el propósito de los diodos en el codificador de teclado? ¿Cuál es la finalidad de las resistencias?

9.9. SÍMBOLOS LÓGICOS CON NOTACIÓN DE DEPENDENCIA

Se presentan dos ejemplos de símbolos con notación de dependencia, según el estándar ANSI/IEEE 91-1984, para los registros de desplazamiento. Se emplean como ejemplos dos registros de desplazamiento en formato de circuito integrado específicos.

Al finalizar esta sección, el lector deberá ser capaz de:

- Entender e interpretar los símbolos lógicos con notación de dependencia para los registros de desplazamiento 74HC164 y 74HC194.

En la Figura 9.39 se presenta el símbolo lógico de un registro de desplazamiento serie con salida paralelo de 8 bits 74HC164. Las entradas de control comunes se indican en el bloque. La entrada de borrado ($\overline{CLR}$) se indica con la letra *R* (RESET) en el interior del bloque. Puesto que no existe prefijo de dependencia para enlazar *R* con el reloj (C1), la función de borrado es asíncrona. La flecha a la derecha de C1 indica el flujo de datos de Q_0 a Q_7. A las entradas *A y B* se les aplica la operación AND, como indica el símbolo AND especificado en el interior del bloque, lo que proporciona la entrada de datos síncrona, 1*D*, en la primera etapa (Q_0). Observe que la dependencia de *D* y *C* se indica mediante el sufijo 1 para *C* y el prefijo 1 para *D*.

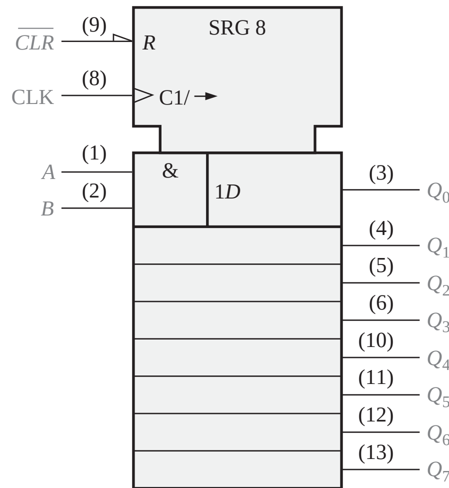


FIGURA 9.39 Símbolo lógico del 74HC164.

La Figura 9.40 es el símbolo lógico para el registro de desplazamiento universal bidireccional de 4 bits 74HC194. Empezando por la parte superior izquierda del bloque, vemos que la entrada $\overline{CLR}$ es activa a nivel BAJO y asíncrona (no hay prefijo de enlace con *C*). Las entradas S_0 y S_1 son entradas de modo que determinan los modos de funcionamiento *desplazamiento a la derecha, desplazamiento a la izquierda* y *carga paralelo*, como indica la designación de dependencia $\frac{0}{3}$ que sigue a la *M*. Esta designación representa los estados binarios de 0, 1, 2 y 3 en las entradas S_0 y S_1. Se establece una dependencia cuando uno de estos dígitos se utiliza como prefijo de otra entrada. El símbolo 1→/2← en la entrada de reloj significa lo siguiente: 1→ indica que se produce un desplazamiento a la derecha (de Q_0 a Q_3) cuando las entradas de modo S_0 y S_1 están en el estado binario 1 ($S_0 = 1$ y $S_1 = 0$), 2← indica que se produce un desplazamiento a la izquierda (de Q_3 a Q_0) cuando las entradas de modo están en el estado binario 2 ($S_0 = 0$ y $S_1 = 1$). La entrada serie para el desplazamiento a la derecha (*SR SER)*, como indica 1,4*D*, es dependiente del modo y dependiente del reloj. Las entradas paralelo (D_0, D_1, D_2 y D_3), como indica 3,4*D*, son dependientes del modo (el prefijo 3 indica modo de carga paralelo) y dependientes del reloj. La entrada serie para desplazamiento a la izquierda (*SL SER)* también es dependiente del modo y de la señal de reloj, como indica 2, 4*D*.

Los cuatro modos del 74HC194 se resumen de la siguiente forma:

No hace nada:	$S_0 = 0$, $S_1 = 0$	(modo 0)
Desplazamiento a la derecha:	$S_0 = 1$, $S_1 = 0$	(modo 1, como en 1,4*D*)
Desplazamiento a la izquierda:	$S_0 = 0$, $S_1 = 1$	(modo 2, como en 2,4*D*)
Carga paralelo:	$S_0 = 1$, $S_1 = 1$	(modo 3, como en 3,4*D*)

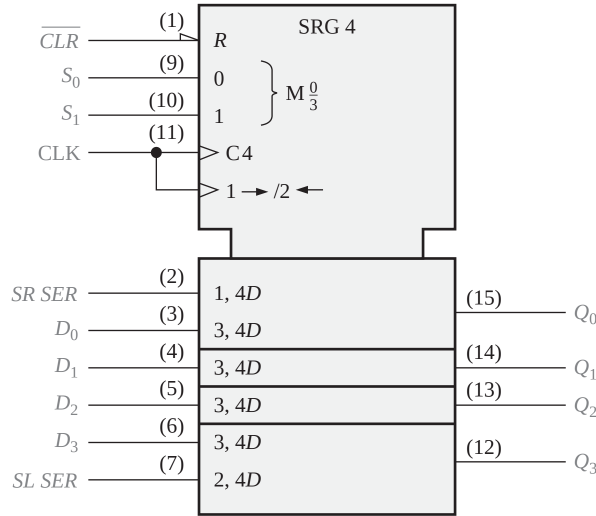


FIGURA 9.40 Símbolo lógico del 74HC194.

REVISIÓN DE LA SECCIÓN 9.9

1. En la Figura 9.43, ¿cuántas entradas son dependientes de las entradas de modo, si se está en el estado 0?
2. ¿Es la carga paralelo síncrona con la señal de reloj?

9.10. LOCALIZACIÓN DE AVERÍAS

En esta sección, vamos a ocuparnos de un método tradicional para la localización de averías en sistemas lógicos secuenciales y otros sistemas digitales más complejos. Dicho método se basa en ejercitar (probar) al circuito bajo prueba con una señal de entrada conocida (estímulo), y luego observar la salida para la secuencia patrón de bits.

Al completar esta sección, el lector deberá ser capaz de:

- Explicar el procedimiento de “ejercitar” un circuito como técnica para la localización de averías.
- Aplicar el procedimiento de “ejercitar” a un convertidor serie-paralelo.

El convertidor de datos serie-paralelo de la Figura 9.33 se usa para ilustrar el procedimiento de “ejercitar” un circuito. El objetivo principal de este procedimiento es forzar a todos los elementos del circuito (flip-flop y puertas) a que pasen por todos sus estados, con el fin de estar seguros de que en ningún estado determinado se produce un fallo. La secuencia patrón de prueba de entrada, en este caso, debe diseñarse para forzar a cada flip-flop de los registros a pasar por ambos estados, hacer que el contador pase por los ocho estados, y comprobar el flip-flop de control, el generador de reloj, el monoestable y la puerta AND.

La secuencia patrón de prueba de entrada que cumple este objetivo para el convertidor de datos serie-paralelo está basada en el formato de datos serie de la Figura 9.34. Se forma mediante el grupo serie de bits de datos 10101010 seguido de otro grupo serie de bits de datos 01010101, como muestra la Figura 9.41. Estas secuencias patrón se generan de forma repetitiva a partir de un generador especial de secuencias de prueba. En la Figura 9.42 se muestra la configuración básica de prueba.

Después de que ambas secuencias patrón han pasado por el circuito bajo prueba, todos los flip-flops de los registros de entrada y salida de datos han pasado por los estados SET y RESET, el contador ha pasado a tra-

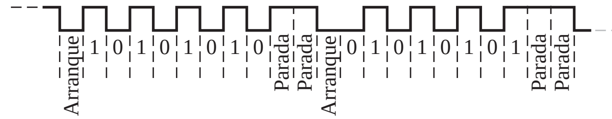


FIGURA 9.41 Secuencia patrón de prueba.

FIGURA 9.42 Configuración de prueba básica para el convertidor de datos serie-paralelo de la Figura 9.33.

vés de su secuencia (una vez para cada secuencia patrón de bits) y todos los demás dispositivos se han comprobado.

Para comprobar el funcionamiento, hay que observar cada una de las salidas de datos paralelo, para las dos secuencias patrón alternativas de 1s y 0s, mientras que las secuencias patrón de prueba de entrada se desplazan repetitivamente por el registro de datos de entrada y luego se cargan en el registro de salida de datos. En la Figura 9.43 se presenta el diagrama de tiempos. Las salidas se pueden observar por parejas utilizando un

osciloscopio de doble traza, o se pueden ver las ocho simultáneamente con un analizador lógico configurado para realizar análisis de tiempos.

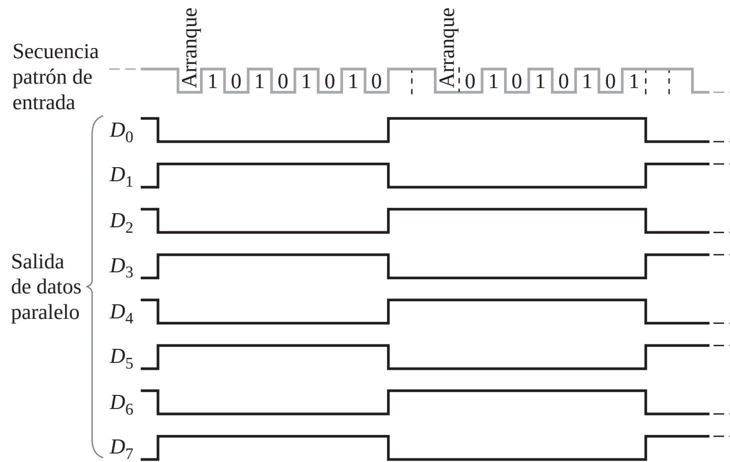


FIGURA 9.43 Salidas correctas del circuito bajo prueba de la Figura 9.42. Se muestran también la secuencia patrón de prueba de entrada.

Si una o más de las salidas del registro de salida de datos es incorrecta, se deben comprobar las salidas del registro de entrada de datos. Si estas salidas son correctas, entonces el problema estará asociado al registro de salida de datos. Compruebe las entradas al registro de salida de datos directamente sobre los pines del CI, para detectar una línea en circuito abierto. Compruebe también que las conexiones de alimentación y a tierra son correctas (buscando la ausencia de ruido en la línea de tierra). Verifique que la línea de carga está a nivel BAJO y que en la entrada de la señal de reloj hay impulsos de reloj de amplitud correcta. Asegúrese de que la conexión al analizador lógico no conecte dos línea de salida, dando lugar a un cortocircuito. Si todas estas pruebas pasan la inspección, entonces, probablemente, es que el registro de salida sea defectuoso. Si las salidas del registro de entrada de datos son también incorrectas, el fallo podría estar en el propio registro de entrada o en cualquier otra parte de la lógica, por lo que será necesaria una investigación adicional para aislar el problema.

CONSEJOS PRÁCTICOS

Cuando se miden señales digitales con un osciloscopio, siempre se debería utilizar el acoplamiento en continua en lugar del acoplamiento en alterna. La razón de que el acoplamiento en alterna no sea mejor para visualizar señales digitales es que el nivel de 0 V de la señal aparecerá en el nivel *medio* de la señal, no en el verdadero nivel de tierra o nivel de 0 V. Es mucho más sencillo encontrar una tierra “flotante” o un nivel lógico incorrecto con el acoplamiento en continua. Si sospecha que hay un punto de tierra en circuito abierto en un circuito digital, incremente la sensibilidad del osciloscopio hasta el máximo posible. Una buena tierra nunca aparecerá con ruido bajo estas condiciones, aunque un circuito abierto probablemente se mostrará con algo de ruido, lo que aparece como una fluctuación aleatoria sobre el nivel de 0 V.

REVISIÓN DE LA SECCIÓN 9.10

1. ¿Cuál es el propósito de proporcionar una entrada de prueba a un circuito lógico secuencial?
2. Generalmente, cuando la señal de salida es incorrecta, ¿cuál es el siguiente paso que se debe dar?



APLICACIÓN A LOS SISTEMAS DIGITALES

En esta aplicación a los sistemas digitales, se va a desarrollar un sistema relativamente sencillo para controlar la seguridad de una sala o de un edificio. El sistema puede programarse mediante un código de seguridad de 4 dígitos, introduciendo los cuatro dígitos de forma secuencial a través de un teclado, en el modo *desactivar* (*Desarm*). Una vez que se ha introducido y almacenado el código de seguridad, el sistema conmuta al modo *activar* (*Arm*). Para desactivar el sistema, es necesario introducir el código correcto de 4 dígitos a través del teclado.

Funcionamiento básico

En la Figura 9.44 se presenta un diagrama de bloques básico del sistema. El sistema lógico está formado por la lógica del código de seguridad y la lógica de memoria. En este capítulo, vamos a centrarnos en la lógica de introducción del código. En el Capítulo 10 se desarrollará la lógica de memoria y se combinarán ambas secciones para formar la lógica del sistema completo.

El conmutador de seguridad $\overline{\text{Arm}}$/Desarm coloca el sistema en modo *activar* o *desactivar*. La programación se realiza colocando primero el sistema en el modo *desactivar* y luego pulsando el conmutador de seguridad *Almacenar* seguido de la tecla correspondiente a cada uno de los cuatro dígitos que hay que introducir. Después de este proceso, la memoria contendrá los códigos BCD de cada uno de los cuatro dígitos del código de seguridad. Cuando el sistema se conmuta al modo *activar*, la señal *SalArm* habilita los sensores del sistema de alarma e ilumina un diodo LED para indicar que el sistema está armado. Para entrar en la sala o en el edificio, es necesario conmutar el sistema al modo *desartivar* e introducir el código de seguridad correcto de cuatro dígitos a través del teclado.

Lógica del código de seguridad

La lógica del código de seguridad controla la activación, desactivación, la programación e introducción de datos. El diagrama lógico básico se muestra en la Figura 9.45. Cuando se activa el sistema por vez primera colocando el conmutador $\overline{\text{Arm}}$/Desarm en la posición *Arm*, el registro de desplazamiento *C* contiene 00010000 de modo que hay un nivel BAJO en *ArmOut*, el cual activa los sensores del sistema, los circuitos de alarma y el indicador ARMADO. También, OSE genera un impulso de reinicialización para el contador de direcciones de la memoria.

Introducción de datos Para desactivar el sistema de modo que pueda entrarse en el área protegida, es necesario introducir el código de cuatro dígitos correcto, que se corresponda con el código almacenado en la memoria. El primer dígito del código de seguridad se introduce a través del teclado. El codificador decimal a BCD genera el código

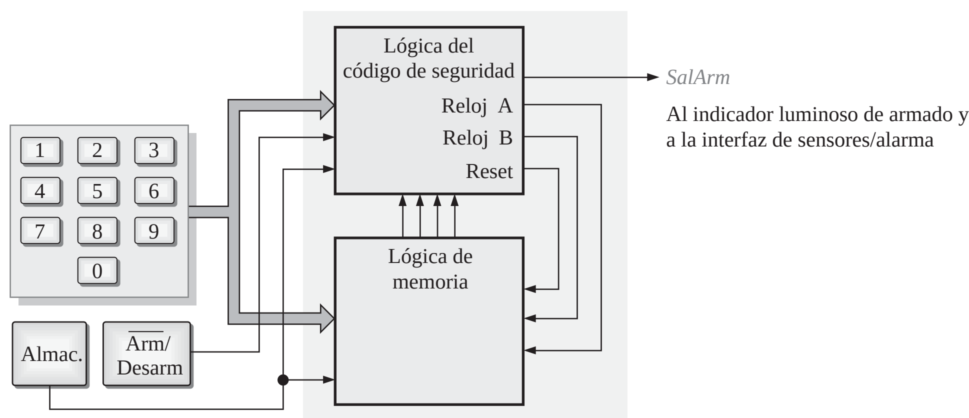


FIGURA 9.44 Diagrama de bloques básico del sistema de seguridad.

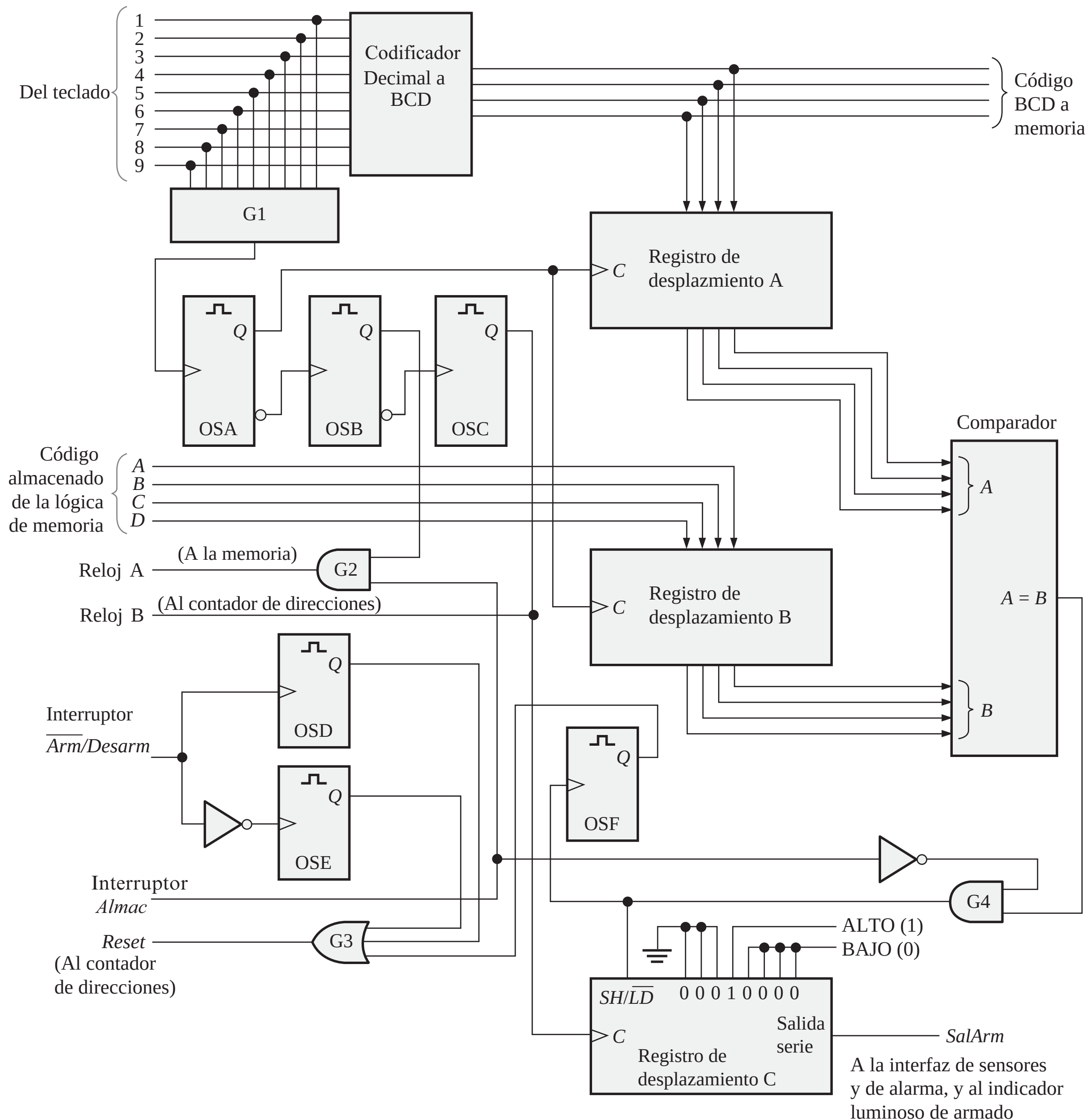


FIGURA 9.45 Diagrama lógico básico de la lógica del código de seguridad.

BCD que representa el dígito que se ha pulsado en el teclado. El monoestable *A* (OSA) se dispara a través de la puerta *G*1 generando un impulso que actúa como señal de reloj para introducir en el registro de desplazamiento A el código BCD de 4 bits procedente del codificador, y para introducir también en el registro de desplazamiento B el código almacenado en la primera dirección de memoria. Una vez que ambos códigos se encuentran en los registros A y B se les aplica como entradas al comparador. Cuando se introduce un código correcto a través del teclado, los 4 bits de las entradas *A* del comparador y los 4 bits de las entradas *B* son iguales, por lo que se produce un nivel ALTO (1) en la salida *A* = *B* del comparador, poniéndose el registro de desplazamiento C en el modo de desplazamiento (*SH*).

El flanco posterior del impulso de salida del circuito OSA dispara OSB, el cual a su vez dispara OSC mediante el flanco posterior de su impulso de salida. La salida de OSC proporciona la señal de reloj *B* al contador de direc-

ciones de memoria y también actúa como reloj para que el registro de desplazamiento *C* desplace el valor 00010000 hacia la derecha, de modo que ahora el registro contendrá 00001000. Puesto que continúa habiendo un 0 (nivel BAJO) en la salida serie *ArmOut*, el sistema permanece armado.

Cuando se introduce a través del teclado el segundo dígito del código correcto, el contenido del registro de desplazamiento C se desplaza para dar 00000100, y el sistema continúa armado. Cuando se introduce el tercer dígito de código a través del teclado, el contenido del registro de desplazamiento C se desplaza para dar 00000010. Cuando se introduce el cuarto y último dígito del código, el contenido del registro de desplazamiento C se desplaza para dar 00000001. Ahora, el nivel ALTO (1) en la salida serie *SalArm* desarma el sistema y permite entrar en la zona protegida.

Si en cualquier momento se introduce un dígito de código incorrecto, la salida del comparador pasa a nivel BAJO, generando un nivel BAJO en la entrada $SH/\overline{LD}$ y dispara OSF para enviar un impulso de reinicialización al contador de direcciones de memoria. El registro de desplazamiento C estará ahora en el modo *carga en paralelo*. OSC se encarga entonces de proporcionar la señal de reloj al registro necesaria para enclavar el código prefijado 00010000 dentro del registro. En este punto, es necesario comenzar de nuevo y reintroducir los cuatro dígitos del código.

Programación Para programar un código de 4 dígitos en el sistema, se coloca el conmutador $\overline{\text{Arm}}$/Desarm en la posición *desactivar*. Esto dispara el monoestable OSD, que envía un impulso de reinicialización a través de *G*3 al contador de direcciones de memoria, haciendo que tome el valor 00, correspondiente a la primera dirección de memoria. El conmutador *Almac* se coloca en la posición *Almacenar,* lo que deshabilita la salida $A = B$ del comparador a través de la puerta *G*4 y habilita la salida de OSB a través de *G*2, para proporcionar un reloj a la memoria durante la introducción del código en la misma.

A continuación, se introduce el primer dígito del código de seguridad deseado a través del teclado. OSA se dispara a través de la puerta *G*1 como resultado del cierre de la tecla y, a su vez, dispara OSB, que genera la señal de reloj *A*, para almacenar el código en la memoria. OSB dispara OSC generando la señal de reloj *B* para el contador de direcciones de memoria haciendo que éste avance hasta la segunda dirección (01). Se introduce el segundo dígito del código a través del teclado y se repite la secuencia descrita para el primer dígito. Después de introducido el cuarto y último código, la memoria contendrá el código de seguridad de cuatro dígitos. Si accidentalmente se introduce un dígito erróneo, es necesario continuar introduciendo los cuatro dígitos o volver a activar el interruptor ALMAC para asegurar que el contador de memoria contenga de nuevo la primera dirección. Una vez realizada la programación, se conmuta el sistema al modo *activar.*

Prácticas de sistemas

- ***Actividad 1*** Describir el propósito del registro de desplazamiento A.
- ***Actividad 2.*** Describir el propósito del registro de desplazamiento B.
- ***Actividad 3.*** Describir el propósito del registro de desplazamiento C.
- ***Actividad 4.*** Describir el propósito del comparador.
- ***Actividad opcional*** Utilizando circuitos integrados lógicos de la familia 74XX y los restantes componentes necesarios, implemente la lógica del código de seguridad descrita en la Figura 9.45. Realice los cambios necesarios para adaptarse a los dispositivos que utilice. Depure y pruebe la lógica y describa los fallos de diseño que encuentre (si es que hay alguno).

RESUMEN

- Los tipos básicos de movimiento de los datos en los registros de desplazamiento se ilustran en la Figura 9.46.
- Los contadores basados en registros de desplazamiento son registros de desplazamiento con realimentación, que disponen de secuencias especiales. Ejemplos de ellos son el contador Johnson y el contador en anillo.
- La secuencia del contador Johnson tiene 2*n* estados, donde *n* es el número de etapas.
- La secuencia del contador en anillo tiene *n* estados.

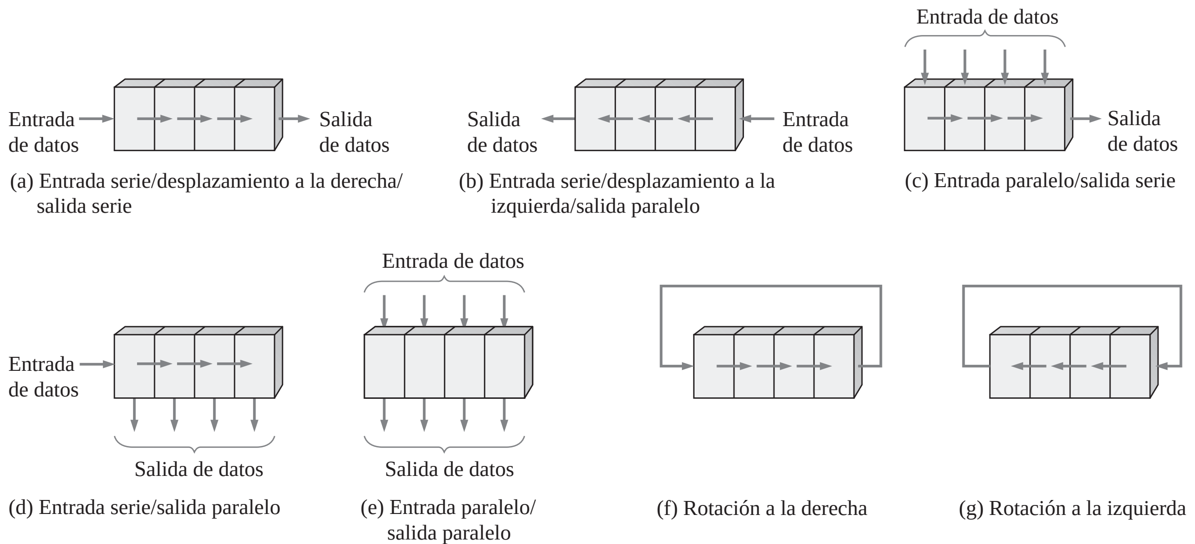


FIGURA 9.46

PALABRAS CLAVE

Las palabras clave y otros términos que se han resaltado en negrita se encuentran en el glosario final del libro.

Bidireccional Que posee dos direcciones. En un registro de desplazamiento bidireccional, los datos almacenados se pueden desplazar a la derecha o a la izquierda.
Carga Introducir datos en un registro de desplazamiento.
Desplazar Mover datos binarios de una etapa a otra dentro de un registro de desplazamiento o de otro dispositivo de almacenamiento; mover datos binarios dentro o fuera de un dispositivo.
Etapa Elemento de almacenamiento en un registro.
Registro Uno o más flip-flops utilizados para almacenar y desplazar datos.

AUTOTEST

Las respuestas se encuentran al final del capítulo.

1. Una etapa de un registro de desplazamiento está formada por:
(a) un latch
(b) un flip-flop
(c) un byte de almacenamiento
(d) cuatro bits de almacenamiento

2. Para desplazar en serie un byte de datos en un registro de desplazamiento, es necesario:
(a) un impulso de reloj
(b) un impulso de carga
(c) ocho impulsos de reloj
(d) un impulso de reloj para cada 1 que contiene el dato

3. Para cargar en paralelo un byte de datos en un registro de desplazamiento con una carga síncrona, es necesario:
(a) un impulso de reloj

(b) un impulso de reloj para cada 1 que contiene el dato

(c) ocho impulsos de reloj

(d) un impulso de reloj para cada 0 que contiene el dato

4. El grupo de bits 10110101 se desplaza en serie (primer bit más a la derecha) a la salida paralelo de 8 bits de un registro de desplazamiento, el cual tiene el estado inicial 11100100. Después de dos impulsos de reloj, el contenido del registro es:

(a) 01011110 **(b)** 10110101 **(c)** 01111001 **(d)** 00101101

5. Con una frecuencia de reloj de 100 kHz, ocho bits se pueden introducir en serie en un registro de desplazamiento en:

(a) 80 μs **(b)** 8 μs **(c)** 80 ms **(d)** 10 μs

6. Con una frecuencia de reloj de 1 MHz, ocho bits se pueden introducir en paralelo en un registro de desplazamiento en:

(a) 8 μs

(b) en un tiempo igual al retardo de propagación de ocho flip-flops

(c) 1 μs

(d) en un tiempo igual al retardo de propagación de un flip-flop

7. Un contador Johnson de módulo 10 requiere:

(a) diez flip-flops **(b)** cuatro flip-flops

(c) cinco flip-flops **(d)** doce flip-flops

8. Un contador en anillo de módulo 10 requiere como mínimo:

(a) diez flip-flops **(b)** cinco flip-flops

(c) cuatro flip-flops **(d)** doce flip-flops

9. Cuando se utiliza un registro de desplazamiento de 8 bits con entrada y salida serie, para obtener un retardo de 24 μs, la frecuencia de reloj debe ser:

(a) 41,67 kHz **(b)** 333 kHz

(c) 125 kHz **(d)** 8 MHz

10. El propósito del contador en anillo del circuito codificador de teclado de la Figura 9.38 es:

(a) aplicar secuencialmente un nivel ALTO a cada fila para detectar la pulsación de una tecla

(b) proporcionar los impulsos de disparo del registro de código de tecla

(c) aplicar secuencialmente un nivel BAJO a cada fila para detectar la pulsación de una tecla

(d) invertir secuencialmente la polarización de los diodos de cada fila.

PROBLEMAS

Las respuestas a los problemas impares se encuentran al final del libro.

SECCIÓN 9.1 Funciones básicas de los registros de desplazamiento

1. ¿Por qué se consideran los registros de desplazamiento dispositivos básicos de memoria?

2. ¿Cuál es la capacidad de almacenamiento de un registro que puede contener dos bytes de datos?

SECCIÓN 9.2 Registros de desplazamiento con entrada y salida serie

3. Para las señales de entrada de datos y de reloj de la Figura 9.47, determinar los estados de cada flip-flop del registro de desplazamiento de la Figura 9.3 y dibujar las formas de onda de salida. Suponer que, inicialmente, el registro contiene todo 1s.

4. Resolver el Problema 3 para las formas de onda de la Figura 9.48.

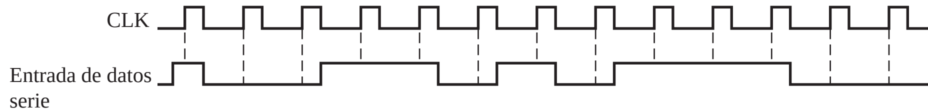


FIGURA 9.47

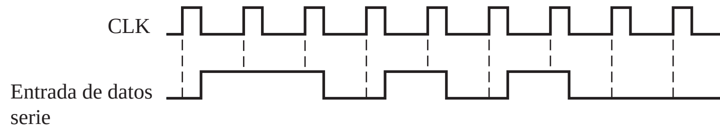


FIGURA 9.48

5. ¿Cuál es el estado del registro de la Figura 9.49 después de cada impulso de reloj, si el estado inicial es 101001111000?

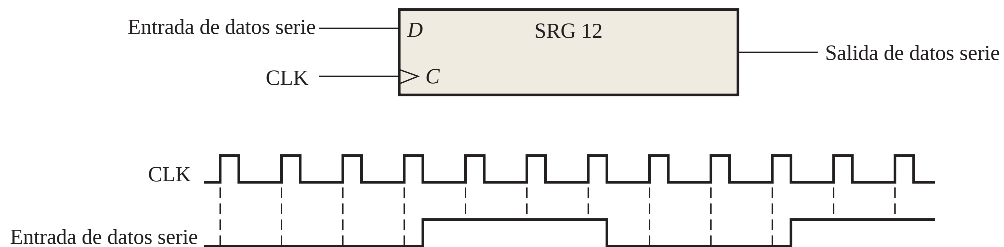


FIGURA 9.49

6. Para el registro de desplazamiento con entrada serie y salida serie, determinar la forma de onda de la salida de datos para la señal de reloj y la entrada de datos de la Figura 9.50. Suponer que, inicialmente, se borra el contenido del registro.

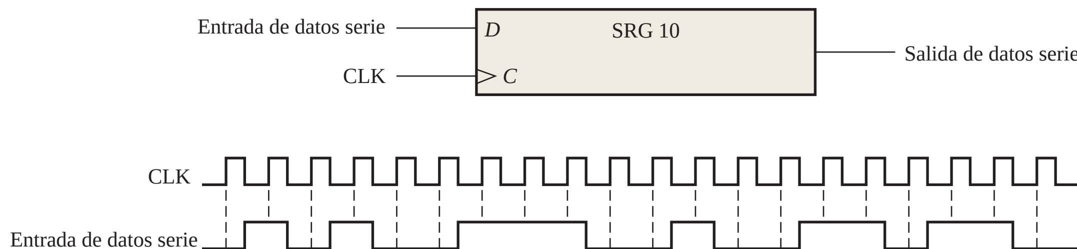


FIGURA 9.50

7. Resolver el Problema 6 para las formas de onda de la Figura 9.51.

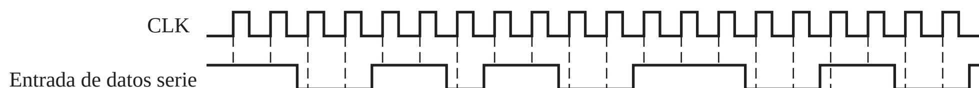


FIGURA 9.51

8. Un registro de desplazamiento de entrada serie-salida paralelo disparado por flanco anterior tiene la forma de onda de la salida de datos mostrada en la Figura 9.52. ¿Qué número binario

se almacena en el registro de 8 bits, si el primer bit de datos que sale (el que está más a la izquierda) es el LSB?

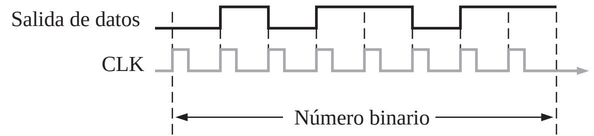


FIGURA 9.52

SECCIÓN 9.3 Registros de desplazamiento con entrada serie y salida paralelo

9. Dibujar un diagrama de tiempos completo que muestre las salidas paralelo del registro de desplazamiento de la Figura 9.8. Utilizar las formas de onda de la Figura 9.50, estando inicialmente borrado el registro.

10. Resolver el Problema 9 para las formas de onda de la Figura 9.51.

11. Desarrollar las salidas Q_0 a Q_7 para un registro de desplazamiento 74HC164, siendo la señal de entrada la mostrada en la Figura 9.53.

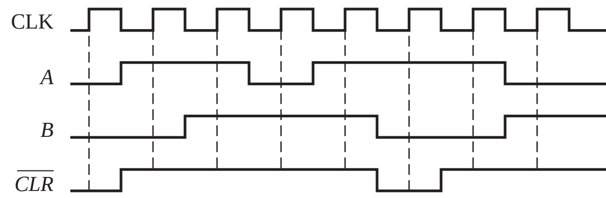
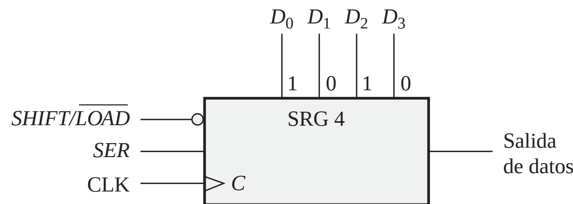


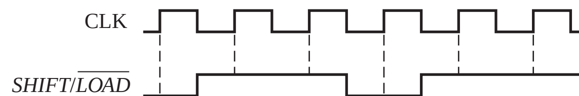
FIGURA 9.53

SECCIÓN 9.4 Registros de desplazamiento con entrada paralelo-salida serie

12. Al registro de desplazamiento de la Figura 9.54(a) se le aplican las señales de entrada $SHIFT/\overline{LOAD}$ y CLK mostradas en la parte (b). La entrada de datos serie (*SER*) está a 0. Las entradas de datos paralelo son $D_0 = 1$, $D_1 = 0$, $D_2 = 1$ y $D_3 = 0$. Dibujar la forma de onda de la salida de datos en función de las entradas.



(a)



(b)

FIGURA 9.54

13. Las formas de onda de la Figura 9.55 se aplican al registro de desplazamiento 74HC165. Todas las entradas paralelo están a 0. Determinar la forma de onda de la salida Q_7.

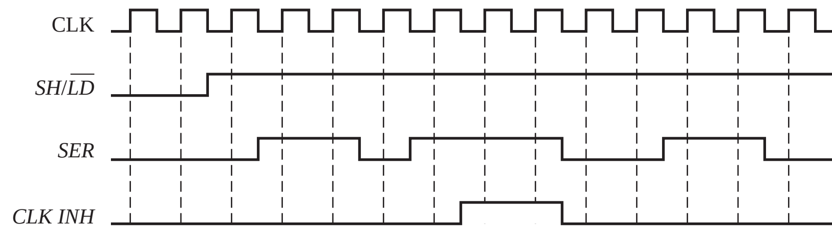


FIGURA 9.55

14. Resolver el Problema 13 si las entradas paralelo están todas a 1.

15. Resolver el Problema 13 si se invierte la entrada *SER*.

SECCIÓN 9.5 Registros de desplazamiento con entrada y salida paralelo

16. Determinar todas las formas de onda de salida *Q* para el registro de desplazamiento de 4 bits 74HC195, cuando las entradas son las indicadas en la Figura 9.56.

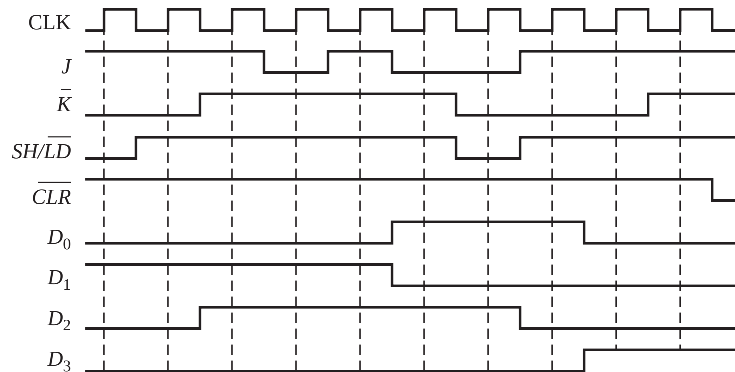


FIGURA 9.56

17. Resolver el Problema 16 si se invierte la entrada $SH/\overline{LD}$ e, inicialmente, se borra el contenido del registro.

18. Utilizar dos registros de desplazamiento 74HC195 para formar un registro de desplazamiento de 8 bits. Mostrar las conexiones que se requieren.

SECCIÓN 9.6 Registros de desplazamiento bidireccionales

19. Para el registro bidireccional de 8 bits de la Figura 9.57, determinar el estado del registro después de cada impulso de reloj para la señal de control $RIGHT/\overline{LEFT}$ dada. Un nivel ALTO en esta entrada activa un desplazamiento a la derecha, y un nivel BAJO lo activa hacia la izquierda. Suponer que, inicialmente, el registro almacena en binario el número decimal setenta y seis, estando en la posición más a la derecha el LSB. La línea de entrada de datos está a nivel BAJO.

20. Resolver el Problema 19 para las señales de la Figura 9.58.

21. Utilizar dos registros de desplazamiento bidireccionales de 4 bits 74HC194, para crear un registro de desplazamiento bidireccional de 8 bits. Indicar las conexiones.

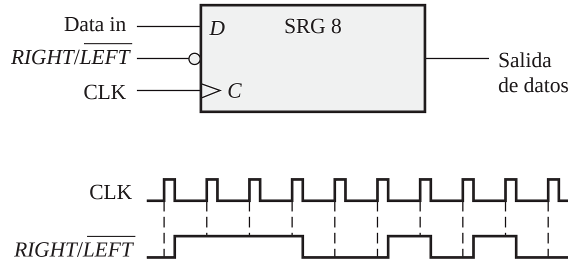


FIGURA 9.57

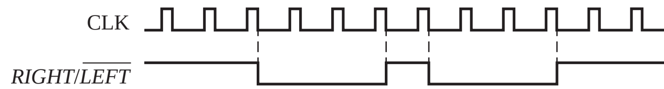


FIGURA 9.58

22. Determinar las salidas Q de un 74HC194 a partir de las entradas mostradas en la Figura 9.59. Las entradas D_0, D_1, D_2 y D_3 están a nivel ALTO.

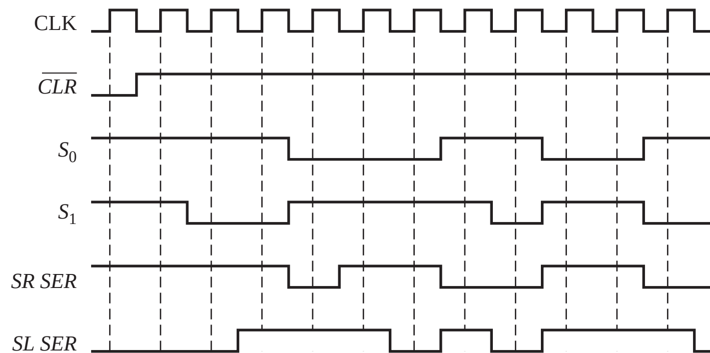


FIGURA 9.59

SECCIÓN 9.7 Contadores basados en registros de desplazamiento

23. ¿Cuántos flip-flops se requieren para implementar cada una de las siguientes configuraciones de un contador Johnson?

(a) módulo 6 **(b)** módulo 10

(c) módulo 14 **(d)** módulo 16

24. Dibujar el diagrama lógico para un contador Johnson de módulo 18. Realizar el diagrama de tiempos y escribir la secuencia en forma de tabla.

25. Para el contador en anillo de la Figura 9.60, dibujar la señal de salida de cada flip-flop en relación con la señal de reloj. Suponer que, inicialmente, FF0 está en estado SET y los demás en estado RESET. Considerar al menos diez impulsos de reloj.

26. A partir de la secuencia patrón mostrada en la Figura 9.60, determinar el contador en anillo e indicar cómo se puede inicializar para generar la señal indicada en la salida Q_9. En el impulso de reloj 16 (CLK16) la secuencia patrón se repite.

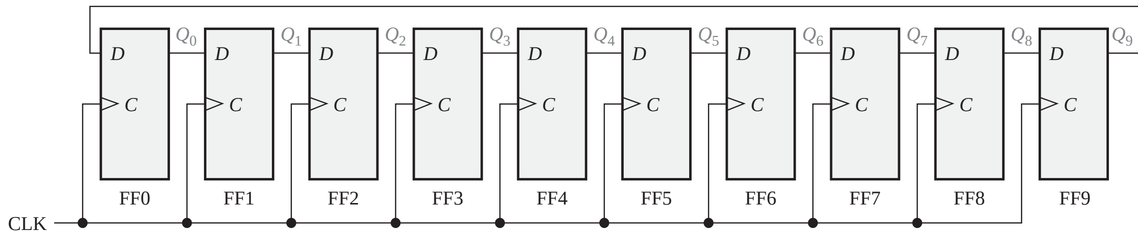


FIGURA 9.60

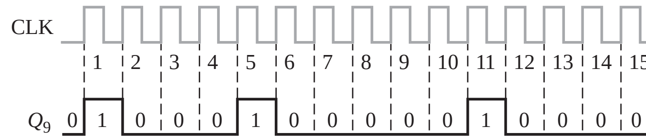


FIGURA 9.61

SECCIÓN 9.8 Aplicaciones de los registros de desplazamiento

27. Utilizar registros de desplazamiento de 4 bits 74HC195 para implementar un contador en anillo de 16 bits. Indicar las conexiones.

28. ¿Cuál es el propósito de la entrada de alimentación $\overline{LOAD}$ de la Figura 9.38?

29. En el esquema de la Figura 9.38 ¿qué ocurre cuando se presionan simultáneamente dos teclas?

SECCIÓN 9.10 Localización de averías

30. A partir de las formas de onda de la Figura 9.62(a), determinar el problema que más probablemente se producirá en el registro mostrado en la parte (b) de la figura.

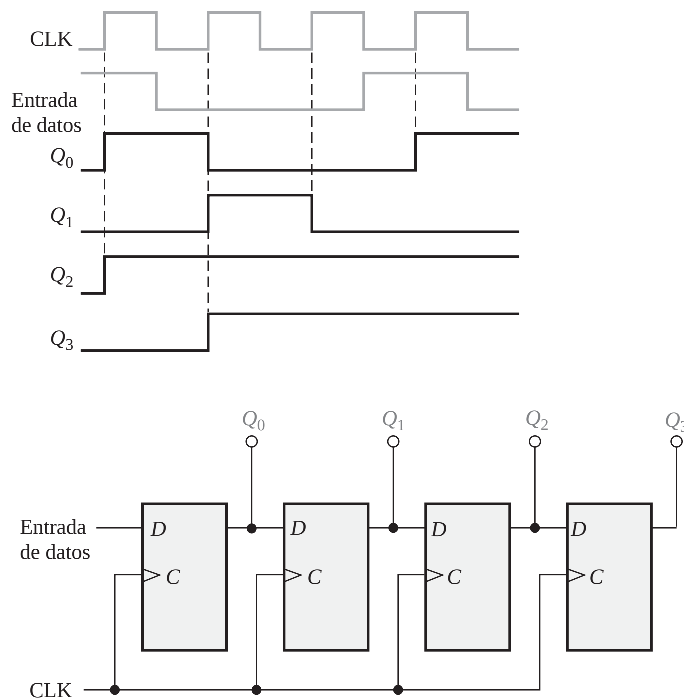


FIGURA 9.62

31. El registro de desplazamiento con entrada paralelo y salida serie de la Figura 9.12 está en el estado en que $Q_0\ Q_1\ Q_2\ Q_3$ = 1001 y en la entrada se carga $D_0\ D_1\ D_2\ D_3$ = 1010. Cuando la entrada $SHIFT/\overline{LOAD}$ está a nivel ALTO, los datos que se muestran en la Figura 9.63 aparecen secuencialmente en la salida. ¿Es correcto este funcionamiento? Si no lo es, ¿cuál es el fallo más probable?

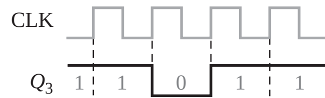


FIGURA 9.63

32. Como hemos visto, en el registro bidireccional de la Figura 9.19, los datos se desplazan hacia la derecha pero no hacia la izquierda. ¿Cuál es el fallo más probable?

33. Para el codificador de teclado de la Figura 9.38, enumerar los posibles fallos para cada uno de los siguientes síntomas:

(a) El estado del registro del código de tecla no cambia cuando se pulsa cualquier tecla.

(b) El estado del registro del código de tecla no cambia cuando se pulsa cualquier tecla de la tercera fila. Cuando se pulsa cualquiera de las restantes teclas se genera un código correcto.

(c) El estado del registro del código de tecla no cambia cuando se pulsa cualquier tecla de la primera columna. Cuando se pulsa cualquiera de las restantes teclas se genera un código correcto.

(d) Cuando se pulsa cualquier tecla de la segunda columna, los tres bits de la izquierda del código de tecla ($Q_0\ Q_1\ Q_2$) son correctos pero los tres bits de la derecha son todos 1s.

34. Desarrollar un procedimiento de prueba para probar el codificador de teclado de la Figura 9.38. Especificar el procedimiento paso a paso, indicando el código de salida del registro de código de tecla que debería observarse en cada paso de las pruebas

35. ¿Qué síntomas se observan si se producen los siguientes fallos en el convertidor serie-paralelo de la Figura 9.33?

(a La salida de la puerta AND se mantiene en estado ALTO.

(b) La salida del generador de reloj se mantiene en estado BAJO

(c) La tercera etapa del registro de entrada de datos se mantiene en estado SET.

(d) La salida de fin de cuenta del contador se mantiene en estado ALTO.



Aplicación a los sistemas digitales

36. ¿Cuál es el propósito principal de la lógica del código de seguridad?

37. Suponer que el código de acceso es 1939. Determinar los estados de los registros de desplazamiento A y C después de haber introducido el segundo dígito correcto.

38. Suponer que el código de acceso es 7646 y que se introduce el código 7645. Determinar los estados de los registros de desplazamiento A y C después de introducir cada uno de los dígitos.



Problemas especiales de diseño

39. Especificar los dispositivos que se pueden utilizar para implementar el convertidor de datos serie-paralelo de la Figura 9.33. Dibujar el diagrama lógico completo mostrando cualquier modificación necesaria para acomodarse a los dispositivos específicos utilizados.

40. Modificar el convertidor serie-paralelo de la Figura 9.33, para conseguir una conversión de 16 bits.
41. Diseñar un convertidor de datos paralelo-serie de 8 bits que produzca el formato de datos de la Figura 9.34. Realizar el diagrama lógico y especificar los dispositivos.
42. Diseñar un circuito de activación de $\overline{LOAD}$ para el codificador de teclado de la Figura 9.38. Este circuito debe generar impulsos de corta duración a nivel BAJO cuando se activa el interruptor de alimentación.
43. Implementar el generador de secuencias patrón de pruebas utilizado en la Figura 9.42 para localizar las averías en el convertidor serie-paralelo.
44. Revisar el sistema de control y de recuento de pastillas introducido en el Capítulo 1. (a) Utilizando los conocimientos adquiridos en este capítulo. implementar los registros A y B en este sistema utilizando circuitos integrados de función fija específicos. (b) Implementar el sistema empleando software de desarrollo.

RESPUESTAS

REVISIONES DE CADA SECCIÓN

SECCIÓN 9.1 Funciones básicas de los registros de desplazamiento

1. Un contador tiene una secuencia específica de estados, pero un registro de desplazamiento no.
2. El almacenamiento y el movimiento de datos son dos funciones de un registro de desplazamiento.

SECCIÓN 9.2 Registros de desplazamiento con entrada y salida serie

1. FF0: entrada de datos a J_0, $\overline{\text{entrada de datos}}$ a K_0; FF1: Q_0 a J_1, $\bar{Q}_0$ a K_1; FF2: Q_1 a J_2, $\bar{Q}_1$ a K_2; FF3: Q_2 a J_3, $\bar{Q}_2$ a K_3.
2. Ocho impulsos de reloj.

SECCIÓN 9.3 Registros de desplazamiento con entrada serie-salida paralelo

1. 0100 después de 2 impulsos de reloj.
2. Se toma la salida serie del flip-flop más a la derecha para operación de salida serie.

SECCIÓN 9.4 Registros de desplazamiento con entrada paralelo-salida serie

1. Cuando $SHIFT/\overline{LOAD}$ está a nivel ALTO, los datos se desplazan a la derecha, un bit por impulso de reloj. Cuando $SHIFT/\overline{LOAD}$ está a nivel BAJO, los datos en las entradas paralelo se cargan en el registro.
2. La operación de carga paralelo es asíncrona, por lo que no depende de la señal de reloj.

SECCIÓN 9.5 Registros de desplazamiento con entrada y salida paralelo

1. Las salidas de datos son 1001
2. $Q_0 = 1$ después de un impulso de reloj

SECCIÓN 9.6 Registros de desplazamiento bidireccionales

1. 1111 después de cinco impulsos de reloj.

SECCIÓN 9.7 Contadores basados en registros de desplazamiento

1. La secuencia del contador Johnson de 8 bits tiene dieciséis estados.
2. Para un contador Johnson de 3 bits: 000, 100, 110, 111, 011, 001, 000

SECCIÓN 9.8 **Aplicaciones de los registros de desplazamiento**

1. 625 exploraciones/segundo
2. $Q_5\, Q_4\, Q_3\, Q_2\, Q_1\, Q_0$ = 011011
3. Los diodos proporcionan caminos unidireccionales para poner las filas a nivel BAJO, y evitar que los niveles ALTOS en las líneas de FILA se conecten a la matriz de interruptores. Las resistencias conectan las líneas de COLUMNA a nivel ALTO.

SECCIÓN 9.9 **Símbolos lógicos con notación de dependencia**

1. Ninguna entrada depende de las entradas de modo que estén en estado 0.
2. Sí, el terminal paralelo es síncrono con el reloj, como indica la etiqueta 4*D*.

SECCIÓN 9.10 **Localización de averías**

1. Se usa una entrada de prueba para que el circuito pase por todos sus estados.
2. Comprobar la entrada de esta parte del circuito. Si la señal en esta entrada es correcta, el fallo queda aislado en la circuitería entre la entrada correcta y la salida incorrecta.

PROBLEMAS RELACIONADOS

9.1. *Véase* la Figura 9.64.

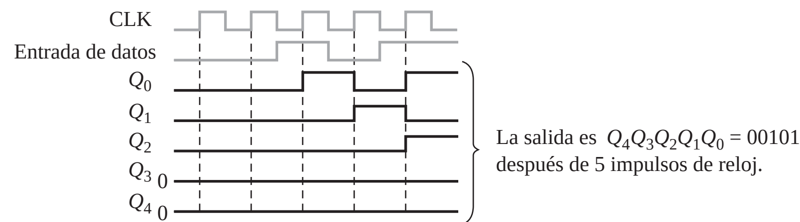


FIGURA 9.64

9.2. El estado del registro después de tres impulsos adicionales de reloj es 0000.

9.3. *Véase* la Figura 9.65.

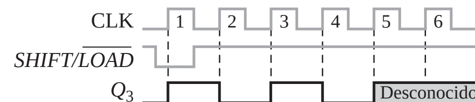


FIGURA 9.65

9.4. *Véase* la Figura 9.66.

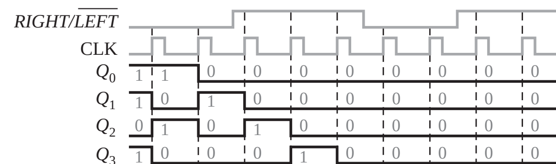


FIGURA 9.66

9.5. *Véase* la Figura 9.67.

9.6. $f = 1/3\ \mu s = 333$ kHz

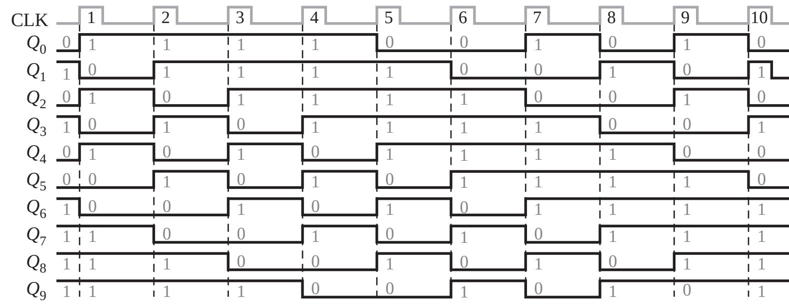


FIGURA 9.67

AUTOTEST

1. (b) **2.** (c) **3.** (a) **4.** (c) **5.** (a)
6. (d) **7.** (c) **8.** (a) **9.** (b) **10.** (c)

10

MEMORIAS Y ALMACENAMIENTO

CONTENIDO DEL CAPÍTULO

OBJETIVOS DEL CAPÍTULO

- Definir las características básicas de las memorias.
- Explicar qué es una memoria RAM y cómo funciona.
- Explicar la diferencia entre RAM estática (SRAM, *Static* RAM) y RAM dinámica (DRAM, *Dynamic* RAM).
- Explicar qué es una memoria ROM y cómo funciona.
- Describir los distintos tipos de memorias PROM.
- Estudiar las características de las memorias flash.



- Describir la expansión de las memorias ROM y RAM para aumentar la longitud y capacidad de palabra.
- Describir los tipos especiales de memorias, como las FIFO y las LIFO
- Describir la organización básica de las cintas y discos magnéticos.
- Describir el funcionamiento básico de los discos magneto-ópticos y de los discos ópticos.
- Describir los métodos básicos para probar las memorias.
- Desarrollar diagramas de flujo para probar las memorias.
- Aplicar las memorias en un sistema digital.

PALABRAS CLAVE

- Byte
- Palabra
- Celda
- Dirección
- Capacidad
- Escritura
- Lectura
- RAM
- ROM
- SRAM
- Bus
- DRAM
- PROM
- EPROM
- Memoria flash
- FIFO
- LIFO
- Disco duro

INTRODUCCIÓN

En el Capítulo 9 se han tratado los registros de desplazamiento, que son un tipo de dispositivos de almacenamiento; de hecho, un registro de desplazamiento es, esencialmente, una memoria a pequeña escala. Los dispositivos de memoria que se cubren en este capítulo se utilizan generalmente para almacenamiento a más largo plazo y de cantidades más grandes de datos de lo que los registros son capaces de permitir.

Las computadoras y otros tipos de sistemas requieren el almacenamiento permanente o semipermanente de un gran número de datos binarios. Los sistemas basados en microprocesador necesitan de los dispositivos de almacenamiento y de las memorias para su funcionamiento, debido a la necesidad de almacenar los programas y mantener los datos generados durante el procesamiento.

En la terminología informática, normalmente el término *memoria* hace referencia a las memorias RAM y ROM y el término *almacenamiento* hace referencia al disco duro, a los discos flexibles y al CD-ROM. En este capítulo se estudian las memorias semiconductoras, y los medios de almacenamiento magnéticos y ópticos.

APLICACIÓN A LOS SISTEMAS DIGITALES

La aplicación a los sistemas digitales del final del capítulo completa el sistema de seguridad del Capítulo 9. Este capítulo se centra en la parte de la lógica de memoria del sistema, que almacena el código de entrada. Una vez que la lógica de memoria se haya desarrollado, se conectará con la lógica de introducción del código del Capítulo 9, para completar el sistema.

10.1 PRINCIPIOS DE LAS MEMORIAS SEMICONDUCTORAS

La memoria es la parte de un sistema que almacena datos binarios en grandes cantidades. Las memorias semiconductoras están formadas por matrices de elementos de almacenamiento que pueden ser *latches* o condensadores.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Explicar cómo almacena una memoria los datos binarios. ■ Exponer la organización básica de una memoria. ■ Describir la operación de escritura. ■ Describir la operación de lectura. ■ Describir la operación de direccionamiento. ■ Explicar qué son las memorias RAM y ROM.

Unidades de datos binarios: bits, bytes, *nibbles* y palabras

Como regla general, las memorias almacenan datos en unidades que tienen de uno a ocho bits. La unidad menor de datos binarios es, como ya sabemos, el **bit**. En muchas aplicaciones, se tratan los datos en unidades de 8 bits, denominadas ***bytes*** o en múltiplos de unidades de 8 bits. El byte se puede dividir en dos unidades de 4 bits, que reciben el nombre de ***nibbles***. Una unidad completa de información se denomina ***palabra*** y está formada, generalmente, por uno o más bytes. Algunas memorias almacenan datos en grupos de 9 bits; un grupo de 9 bits consta de un byte más un bit de paridad.



NOTAS INFORMÁTICAS

La definición general de *palabra* dice que una palabra es una unidad completa de información, consistente en una unidad de datos binarios. Cuando se aplica a las instrucciones de una computadora, una palabra se define de forma más específica como dos bytes (16 bits). Como parte muy importante del lenguaje ensamblador utilizado por las computadoras, la directiva DW (*Define Word*) significa que se definen datos en unidades de 16 bits. Esta definición es independiente del microprocesador o del tamaño de su bus de datos. El lenguaje ensamblador también permite definiciones de bytes (8 bits) con la directiva DB, de palabras dobles (32 bits) con la instrucción DD y de palabras cuádruples (64 bits) con la instrucción QD.

Matriz de memoria semiconductora básica

Cada elemento de almacenamiento en una memoria puede almacenar un 1 o un 0 y se denomina ***celda***. Las memorias están formadas por matrices de celdas, como se ilustra en la Figura 10.1, en la que se utilizan 64

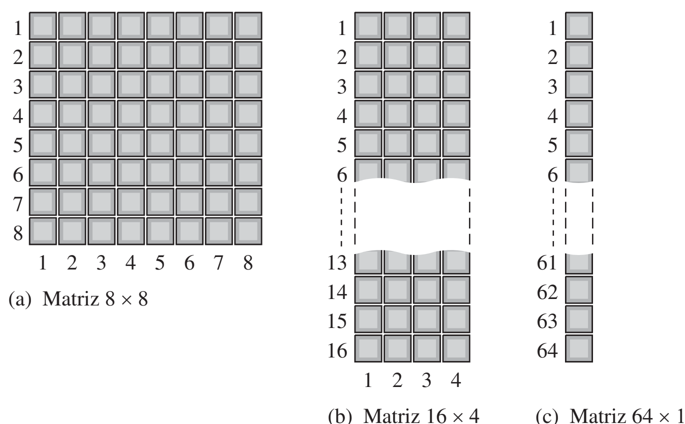


FIGURA 10.1 Matriz de almacenamiento de 64 celdas, organizada de tres formas diferentes.

celdas a modo de ejemplo. Cada bloque de la **matriz de memoria** representa una celda de almacenamiento y su situación se puede especificar mediante una fila y una columna.

La matriz de 64 celdas se puede organizar de muchas maneras en función de las unidades de datos. La Figura 10.1(a) muestra una matriz de 8 × 8, que se puede entender como una memoria de 64 bits o como una memoria de 8 bytes. La parte (b) nos muestra una matriz de 16 × 4, que es una memoria de 16 *nibbles* y la parte (c) presenta una matriz de 64 × 1 que es una memoria de 64 bits. Una memoria se identifica mediante el número de palabras que puede almacenar, multiplicado por el tamaño de la palabra. Por ejemplo, una memoria de 16k × 8 puede almacenar 16.384 palabras de ocho bits. La incoherencia en la expresión anterior es común en la terminología de las memorias. En realidad, el número de palabras es siempre una potencia de 2 que, en este caso, es 2^{14} = 16.384. Sin embargo, es una práctica común expresar cada número redondeado al millar más próximo, en este caso 16k.

Dirección y capacidad de las memorias

La posición de una unidad de datos en una matriz de memoria se denomina ***dirección***. Por ejemplo, en la Figura 10.2(a), la dirección de un bit en la matriz de dos dimensiones se especifica mediante la fila y columna en que está, tal como se muestra. En la Figura 10.2(b), la dirección de un byte se especifica únicamente mediante la fila. Como puede ver, la dirección depende de cómo se organice la memoria en unidades de datos. Las computadoras personales disponen de memorias organizadas en bytes. Esto significa que el grupo más pequeño de bits que se puede direccionar es ocho.

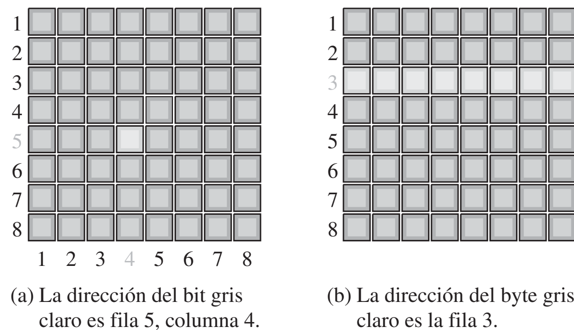


FIGURA 10.2 Ejemplos de direcciones de memoria en una matriz bidimensional.

En la Figura 10.3, la dirección de un byte en una matriz tridimensional se especifica mediante la fila y la columna correspondientes. En este caso, el grupo más pequeño de bits al que se puede acceder es ocho.

La ***capacidad*** de una memoria es el número total de unidades de datos que puede almacenar. Por ejemplo, en la matriz de memoria organizada en bits de la Figura 10.2(a), la capacidad total es de 64 bits. En la matriz de memoria organizada en bytes de la Figura 10.2(b), la capacidad es de 8 bytes, que es lo mismo que 64 bits. En la Figura 10.3, la capacidad es de 64 bytes. Típicamente, las memorias de computadora disponen de 256 MB (MB es megabyte), o más, de memoria interna.

Operaciones básicas de las memorias

Puesto que una memoria almacena datos binarios, los datos deben introducirse en la memoria y deben poder recuperarse cuando se necesiten. La operación de ***escritura*** coloca los datos en una posición específica de la memoria y la operación de ***lectura*** extrae los datos de una dirección específica de memoria. La operación de

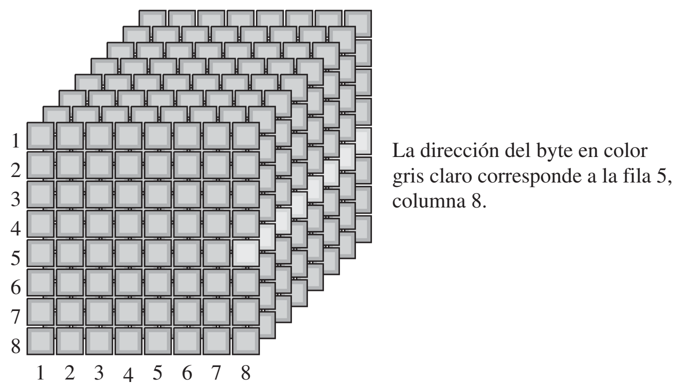
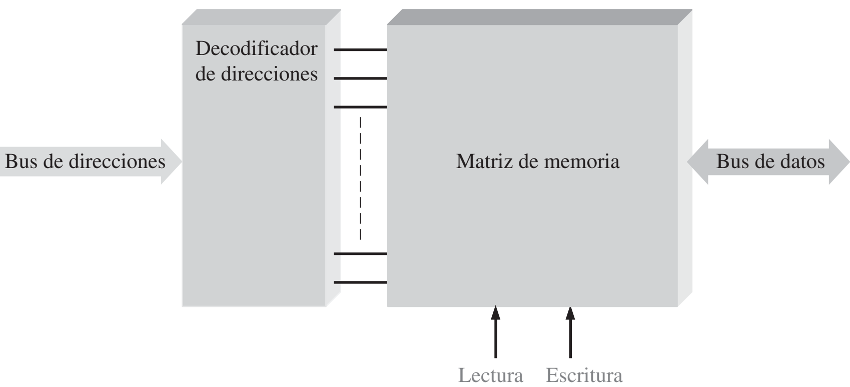
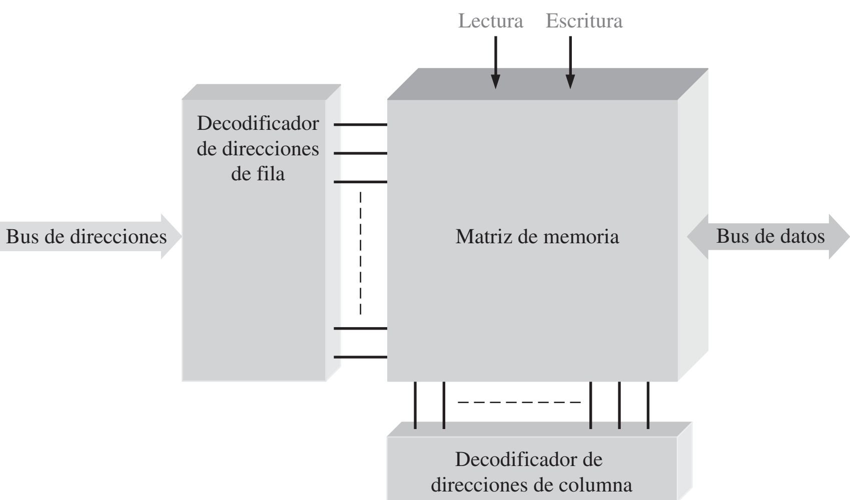


FIGURA 10.3 Ejemplo de dirección de memoria en una matriz de tres dimensiones.



(a) Matriz de memoria bidimensional



(b) Matriz de memoria de tres dimensiones

FIGURA 10.4 Diagramas de bloques de una memoria de 2 dimensiones y de otra de 3 dimensiones, mostrando el bus de direcciones, el decodificador de direcciones, el bus de datos bidireccional y las entradas de lectura/escritura.

direccionamiento, que forma parte tanto de la operación de lectura como de la de escritura, selecciona la dirección de memoria específica.

Las unidades de datos se introducen en la memoria durante la operación de escritura y se extraen de la memoria durante la operación de lectura a través de un conjunto de líneas que se denominan *bus de datos*. Como se indica en la Figura 10.4, el bus de datos es bidireccional, lo que significa que los datos pueden ir en cualquiera de las dos direcciones (hacia la memoria o desde la memoria). En el caso de una memoria organizada en bytes, el bus de datos tiene al menos ocho líneas, de manera que los ocho bits de una dirección seleccionada se transmiten en paralelo. En una operación de escritura o de lectura, se selecciona una dirección introduciendo un código binario, que representa la dirección deseada, en un conjunto de líneas denominado *bus de direcciones*. El código de dirección se decodifica internamente y de esa forma se selecciona la dirección adecuada. En el caso de la matriz de memoria de 3 dimensiones de la Figura 10.4(b) se usan dos decodificadores, uno para las filas y otro para las columnas. El número de líneas del bus de direcciones depende de la capacidad de la memoria. Por ejemplo, un código de dirección de 15 bits puede seleccionar 32.768 posiciones (2^{15}) en la memoria; un código de dirección de 16 bits puede seleccionar 65.536 (2^{16}) posiciones de memoria, etc. En las computadoras personales, un bus de direcciones de 32 bits puede seleccionar 4.294.967.296 (2^{32}) posiciones, lo que se expresa como 4 G.

La operación de escritura. En la Figura 10.5 se muestra la operación de escritura simplificada. Para almacenar un byte de datos en memoria, se introduce en el bus de direcciones un código que se encuentra almacenado en el registro de direcciones. Una vez que el código de dirección está ya en el bus, el decodificador de direcciones decodifica la dirección y selecciona la posición de memoria especificada. La memoria recibe entonces una orden de escritura y los datos almacenados en el registro de datos se introducen en el bus de datos, y se almacenan en la dirección de memoria especificada, completándose así la operación de escritura. Cuando se

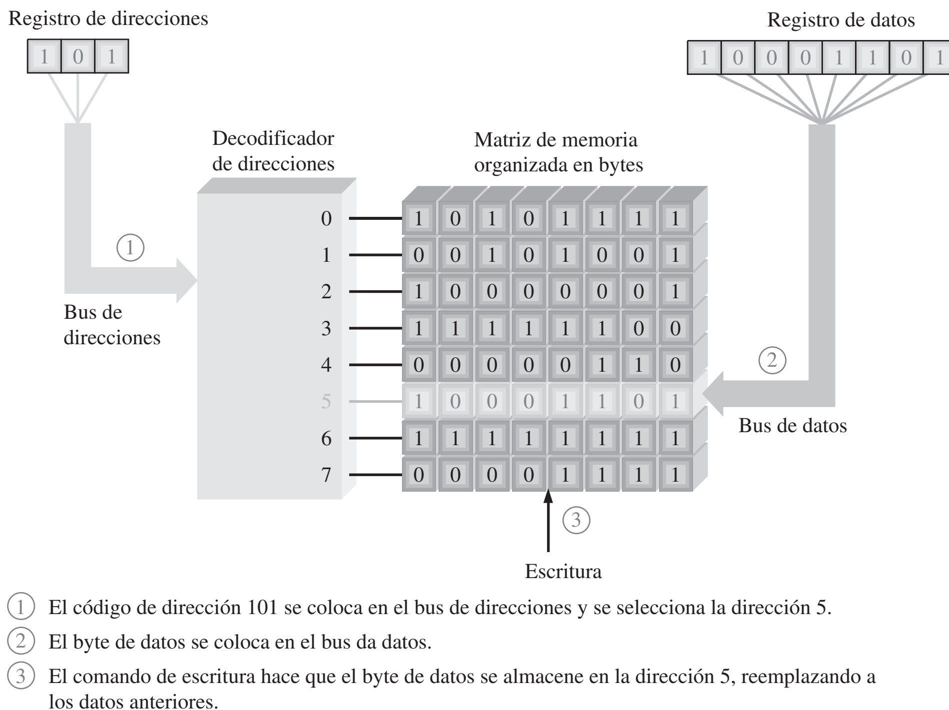


FIGURA 10.5 Ilustración de la operación de escritura.

escribe un nuevo byte de datos en una dirección de memoria, se sobrescribe y destruye el byte de datos actualmente almacenado en esa dirección.

La operación de lectura. En la Figura 10.6 se muestra la operación de lectura simplificada. De nuevo, se introduce en el bus de direcciones un código almacenado en el registro de direcciones. Una vez que el código de dirección se encuentra en el bus, el decodificador de direcciones decodifica la dirección y selecciona la posición especificada de la memoria. La memoria recibe entonces una orden de lectura, y una "copia" del byte de datos almacenado en la dirección de memoria seleccionada se introduce en el bus de datos y se carga en el registro de datos, finalizando así la operación de lectura. Cuando se lee un byte de datos de una dirección de memoria, éste sigue almacenado en dicha dirección. Esto se denomina *lectura no destructiva*.

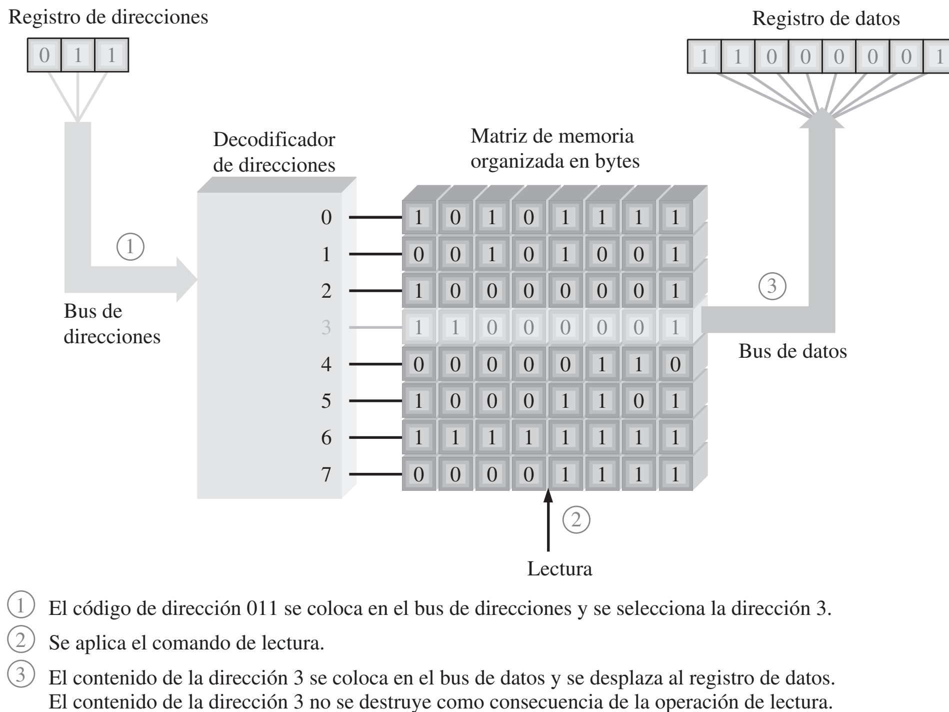


FIGURA 10.6 Ilustración de la operación de lectura.

Las memorias RAM y ROM

Las dos principales categorías de memorias semiconductoras son las memorias RAM y ROM. La memoria ***RAM*** (*Random-Access Memory*, memoria de acceso aleatorio) es un tipo de memoria en la que se tarda lo mismo en acceder a cualquier dirección de memoria y éstas se pueden seleccionar en cualquier orden, tanto en una operación de lectura como de escritura. Todas las RAM poseen la capacidad de *lectura* y *escritura*. Debido a que las memorias RAM pierden los datos almacenados cuando se desconecta la alimentación, reciben el nombre de memorias **volátiles**.

La memoria **ROM** (*Read-Only Memory*, memoria de sólo lectura) es un tipo de memoria en la que los datos se almacenan de forma permanente o semipermanente. Los datos se pueden leer de una ROM, pero no existe la operación de escritura como en las RAM. La ROM, al igual que la RAM, es una memoria de acceso aleatorio pero, tradicionalmente, el término RAM se reserva para las memorias de acceso aleatorio de *lec-*

tura/escritura. En este capítulo veremos varios tipos de memorias RAM y ROM. Debido a que las ROM mantienen los datos almacenados incluso si se desconecta la alimentación, reciben el nombre de memorias **no volátiles**.

REVISIÓN DE LA SECCIÓN 10.1

Las respuestas se encuentran al final del capítulo

1. ¿Cuál es la unidad más pequeña de datos que se puede almacenar en una memoria?
2. ¿Cuál es la capacidad en bits de una memoria que puede almacenar 256 bytes de datos?
3. ¿En qué consiste la operación de escritura?
4. ¿En qué consiste la operación de lectura?
5. ¿Cómo se localiza una determinada unidad de datos en una memoria?
6. Describir las diferencias entre una RAM y una ROM.

10.2 MEMORIAS DE ACCESO ALEATORIO (RAM)

Las RAM son memorias de lectura-escritura en las que los datos se pueden escribir o leer en cualquier dirección seleccionada en cualquier secuencia. Cuando se escriben los datos en una determinada dirección de la RAM, los datos almacenados previamente son reemplazados por la nueva unidad de datos. Cuando una unidad de datos se lee de una determinada dirección de la RAM, los datos de esa dirección permanecen almacenados y no son borrados por la operación de lectura. Esta operación no destructiva de lectura se puede entender como una copia del contenido de una dirección, dejando dicho contenido intacto. La RAM se utiliza habitualmente para almacenamiento de datos a corto plazo, ya que no puede conservar los datos almacenados cuando se desconecta la alimentación.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Nombrar las dos categorías de RAM. ■ Explicar qué es una SRAM. ■ Describir la celda de almacenamiento de una SRAM. ■ Explicar la diferencia entre una SRAM asíncrona y una SRAM de ráfaga síncrona. ■ Explicar qué es una DRAM. ■ Describir la celda de almacenamiento de una DRAM. ■ Explicar los tipos de DRAM. ■ Comparar la SRAM con la DRAM.

La familia de memorias RAM

Las dos categorías de memorias RAM son la *RAM estática* (SRAM) y la *RAM dinámica* (DRAM). Las RAM estáticas utilizan generalmente *latches* como elementos de almacenamiento y, por tanto, pueden almacenar datos de forma indefinida *siempre que se aplique una alimentación continua.* Las RAM dinámicas utilizan condensadores como elemento de almacenamiento y no pueden mantener los datos mucho tiempo sin recargar los condensadores mediante el proceso de **refresco**. Tanto las SRAM como las DRAM perderán los datos cuando se elimine la alimentación continua, por lo que se clasifican como memorias volátiles.

Los datos pueden leerse mucho más rápidamente en una SRAM que en una DRAM. Sin embargo, las DRAM pueden almacenar muchos más datos que las SRAM para un tamaño físico y coste dados, ya que la celda de las DRAM es mucho más sencilla y se pueden incluir muchas más celdas en un área determinada que en una memoria SRAM.

Los tipos básicos de memorias SRAM son las memorias *SRAM asíncronas* y las *SRAM síncronas de ráfaga*. Los tipos básicos de DRAM son la *DRAM con modo página rápido (Fast Page Mode, FPM DRAM)*, la *DRAM con salida de datos extendida (Extended Data Output, EDO DRAM)*, la *DRAM con salida de datos extendida en ráfaga (Burst Extended Data Output, BEDO DRAM)* y la *DRAM síncrona (Synchronous, SDRAM)*. Todas ellas se muestran en la Figura 10.7.

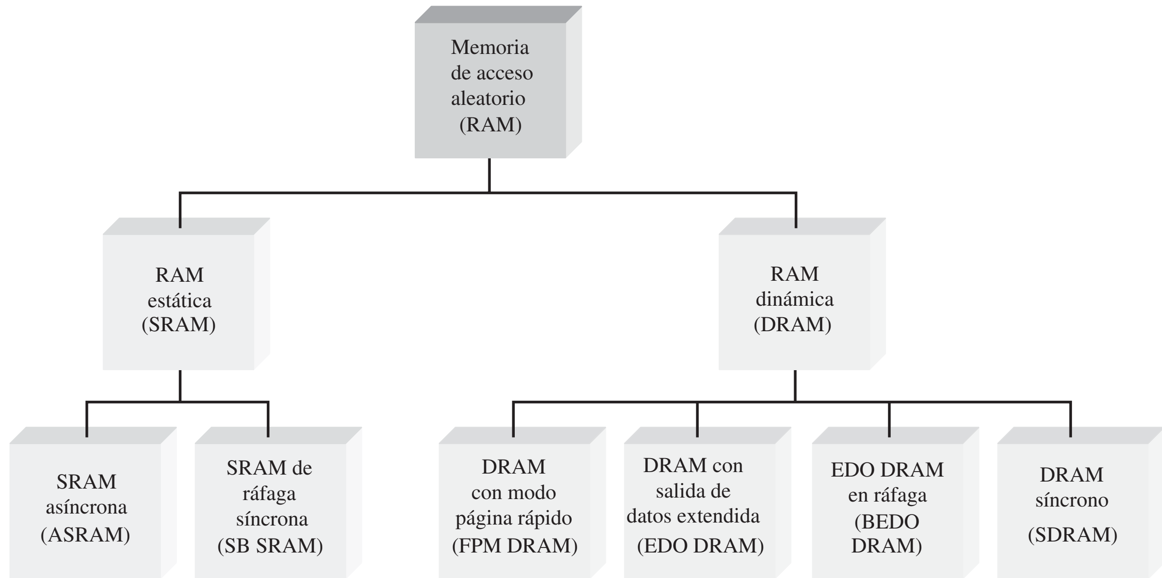


FIGURA 10.7 La familia de memorias RAM.

RAM estática (SRAM)

Celda de memoria. Todas las RAM estáticas se caracterizan por las celdas de memoria *latch*. Cuando se aplica alimentación continua a una celda de **memoria estática** se puede mantener un estado 1 o 0 indefinidamente. Si se retira la alimentación, el bit de datos almacenado se perderá.

La Figura 10.8 muestra una celda de memoria de tipo *latch* para la ***SRAM***. La celda se selecciona mediante un nivel activo en la línea Selección de bit y un bit de datos (1 o 0) se escribe en la celda colocándolo en la línea Entrada de datos. Un bit de datos se puede leer extrayéndolo de la línea Salida de datos.

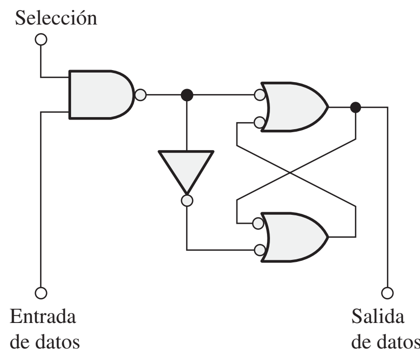


FIGURA 10.8 Celda típica de memoria *latch* de una SRAM.

Matriz básica de celdas de memoria estáticas. Las celdas de almacenamiento en una SRAM se organizan en filas y columnas, como se ilustra en la Figura 10.9 para el caso de una matriz $n \times 4$. Todas las celdas de una misma fila comparten la misma línea Seleccionar Fila. Cada conjunto de líneas Entrada de datos y Salida de datos

van a cada celda situada en una determinada columna y se conectan a una única línea de datos, que sirve como entrada y salida (E/S datos), a través de los buffers de entrada y salida de datos.

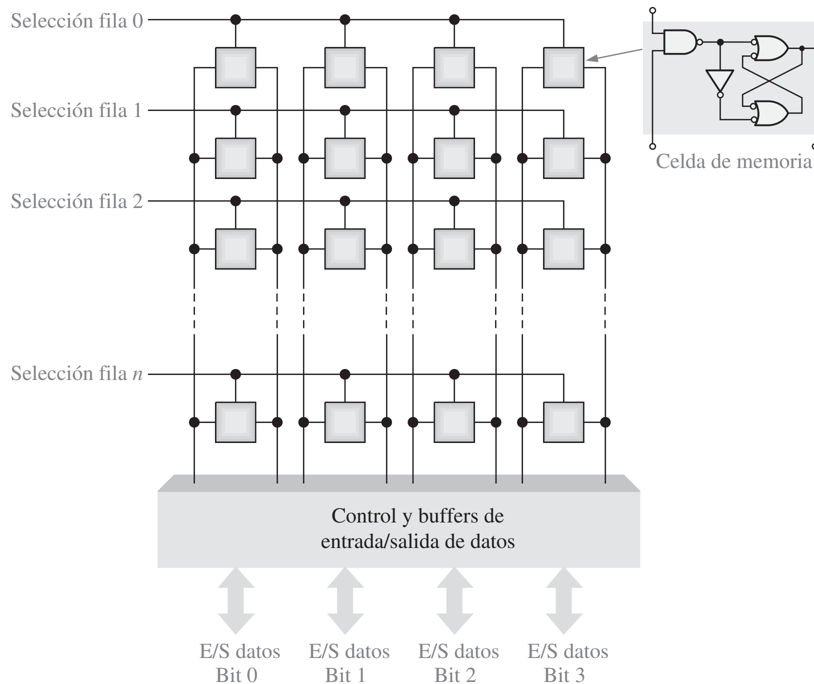


FIGURA 10.9 Matriz SRAM básica.

Para escribir una unidad de datos, en este caso un grupo de 4 bits, en una fila de celdas determinada de la matriz de memoria, la línea Selección Fila se pone en su estado activo y los cuatro bits de datos se colocan en las líneas de entrada de datos. La línea de escritura (*write*) se pone entonces en estado activo, lo que da lugar a que cada bit de datos se almacene en una celda seleccionada en la columna asociada. Para leer una unidad de datos, se pone en estado activo la línea de lectura (*read*), lo que hace que los cuatro bits de datos almacenados en la fila seleccionada aparezcan en las líneas de E/S de datos.

Organización de la SRAM asíncrona básica

Una SRAM asíncrona es aquélla en la que su funcionamiento no está sincronizado con un reloj de sistema. Para ilustrar la organización general de una SRAM, vamos a utilizar una memoria de 32 K × 8 bits. En la Figura 10.10 se muestra el símbolo lógico de esta memoria.

En el modo de lectura (READ), los 8 bits de datos que se almacenan en una dirección determinada aparecen en las líneas de salida de datos. En el modo de escritura (WRITE), los 8 bits de datos que se aplican a las líneas de entrada de datos se almacenan en la dirección seleccionada. Las líneas de entrada y salida de datos (E/S_1 a E/S_8) son las mismas líneas. Durante la operación de lectura, éstas actúan como líneas de salida (S_1 a S_8) y durante la operación de escritura actúan como líneas de entrada (E_1 a E_8).

Salidas tri-estado y buses. Los *buffers* tri-estado en una memoria permiten que las líneas de datos actúen como líneas de entrada o salida y conectan la memoria con el bus de datos en una computadora. Estos buffers tienen tres posibles estados de salida: ALTO (1), BAJO (0) y ALTA-Z (alta impedancia, abierto). Las salidas

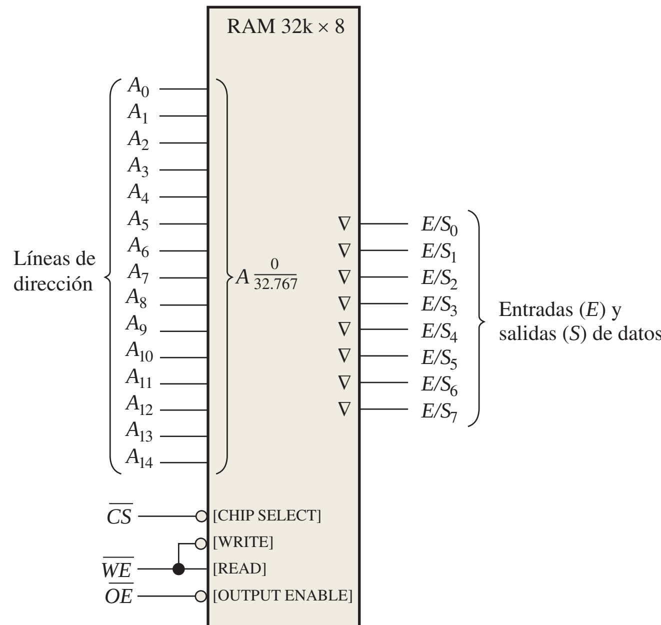


FIGURA 10.10 Diagrama lógico de una SRAM asíncrona de 32 k × 8.

tri-estado se indican en los símbolos lógicos mediante un pequeño triángulo invertido (∇), como se muestra en la Figura 10.9, y se utilizan por compatibilidad con las estructuras de bus, como las que se encuentran en los sistemas basados en microprocesador.

Físicamente, un bus es un conjunto de caminos conductores que sirven para interconectar dos o más componentes funcionales de un sistema o de varios sistemas diferentes. Eléctricamente, un bus es una colección de señales y de niveles de tensión y/o corriente específicos que permiten a los distintos dispositivos conectados al bus comunicarse y funcionar correctamente.

Por ejemplo, un microprocesador se conecta a las memorias y dispositivos de entrada/salida mediante determinadas estructuras de bus. Un bus de direcciones permite al microprocesador direccionar las memorias, y el bus de datos sirve para la transferencia de datos entre el microprocesador, las memorias y los dispositivos de entrada/salida, tales como monitores, impresoras, teclados y modems. El bus de control permite al microprocesador controlar la temporización y la transferencia de datos entre los distintos componentes.

Matriz de memoria. Los chips SRAM se pueden organizar en bits, *nibbles* (4 bits), bytes (8 bits) o múltiplos de bytes (16, 24 o 32 bits).

La Figura 10.11 muestra la organización de una SRAM típica de 32 k × 8. La matriz de celdas de memoria está organizada en 256 filas y 128 columnas, cada una de 8 bits, como se muestra en la parte (a) de la figura. En la práctica, dispone de 2^{15} = 32.768 direcciones, y cada dirección contiene 8 bits. La capacidad de esta memoria ejemplo es entonces de 32.768 bytes (lo que normalmente se expresa como 32 kB).

La SRAM de la Figura 10.11(b) trabaja del siguiente modo. En primer lugar, la entrada de habilitación del chip, $\overline{CS}$, debe estar a nivel BAJO para que la memoria funcione. Ocho de las quince líneas de dirección se decodifican en el decodificador de filas, de modo que se selecciona una de las 256 filas. Las restantes siete líneas de dirección las decodifica el decodificador de columnas, de modo que se selecciona una de las 128 columnas de 8 bits.

Lectura. En el modo lectura (READ), la entrada de habilitación de escritura $\overline{WE}$ está a nivel ALTO y la salida de habilitación $\overline{OE}$ está a nivel BAJO. La puerta G_1 desactiva los buffers de entrada, y la puerta G_2 activa

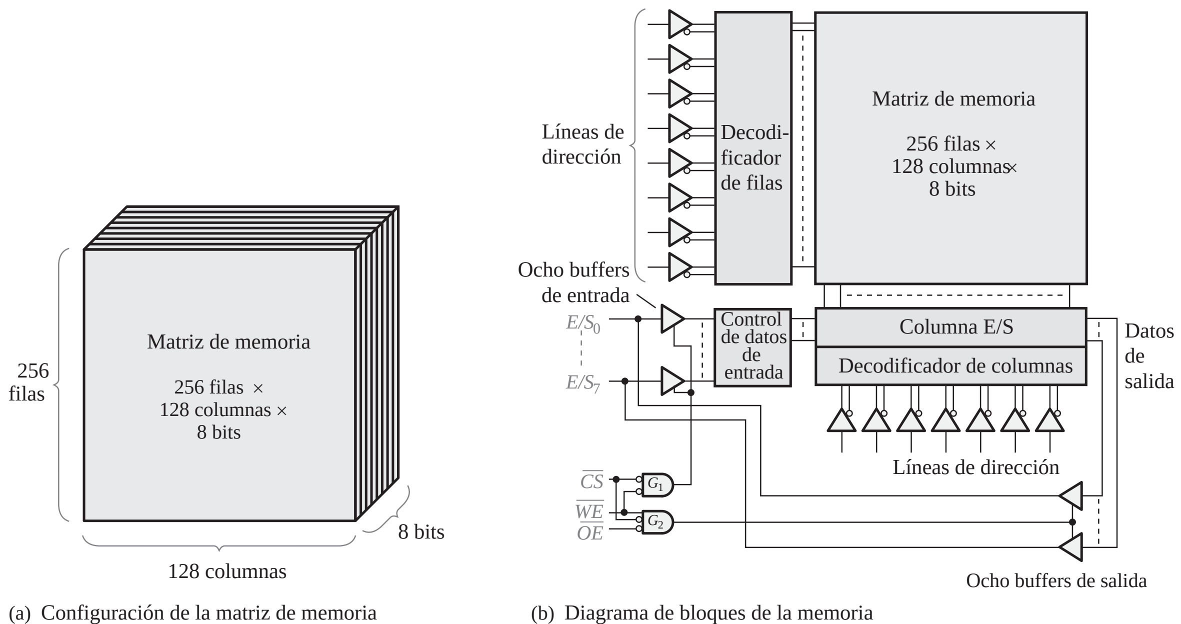


FIGURA 10.11 Organización básica de una SRAM asíncrona de 32k × 8.

los buffers de tres estados de salida de las columnas. Por tanto, los ocho bits de datos almacenados en la dirección seleccionada se llevan a través de las E/S de las columnas hasta las líneas de datos (E/S_1 a E/S_8), que actúan como líneas de salida de datos.

Escritura. En el modo escritura (WRITE), $\overline{WE}$ está a nivel BAJO y $\overline{OE}$ está a nivel ALTO. La puerta G_1 activa los buffers de entrada, y la puerta G_2 desactiva los buffers de salida. Por tanto, los ocho bits de datos de entrada de las líneas de datos se llevan a través del control de datos de entrada y de la E/S de columna a la dirección seleccionada, y se almacenan.

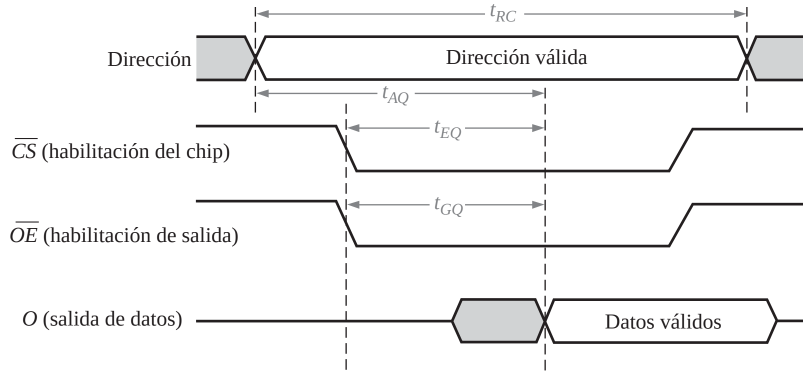
Ciclos de lectura y escritura. La Figura 10.12 muestra un diagrama de tiempos típico para un ciclo de lectura y un ciclo de escritura de una memoria. En la parte (a) se presenta el ciclo de lectura; se aplica un código de dirección válido a las líneas de dirección durante un intervalo de tiempo especificado, que se denomina *período del ciclo de lectura, t_{RC}*. A continuación, las entradas de habilitación del chip ($\overline{CS}$) y de habilitación de salida ($\overline{OE}$) pasan a nivel BAJO. Un intervalo de tiempo después de que la entrada $\overline{OE}$ haya pasado a nivel BAJO, un byte de datos válido procedente de la dirección seleccionada se presenta en las líneas de datos. Este intervalo de tiempo se denomina *tiempo de acceso de la habilitación de salida, t_{GQ}.* Existen otros dos tiempos de acceso en el ciclo de lectura: el *tiempo de acceso de dirección, t_{AQ}*, que se mide desde el principio de una dirección valida hasta que los datos válidos aparecen en las líneas de datos, y el *tiempo de acceso de la habilitación del chip, t_{EQ}*, que se mide desde la transición de nivel ALTO a nivel BAJO de $\overline{CS}$ hasta que los datos válidos aparecen en las líneas de datos.

En cada ciclo de lectura, se lee de la memoria una unidad de datos, en este caso un byte.

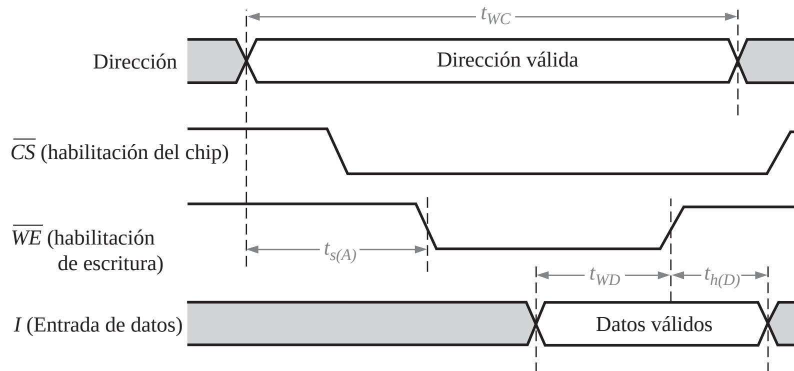
La Figura 10.12(b) muestra el ciclo de escritura. Un código de dirección válido se aplica a las líneas de dirección durante un intervalo de tiempo especificado que se denomina *período del ciclo de escritura, t_{WC}*. A continuación, las entradas de habilitación del chip ($\overline{CS}$) y de habilitación de escritura ($\overline{WE}$) pasan a nivel BAJO. El intervalo de tiempo requerido desde el inicio de una dirección válida hasta que la entrada $\overline{WE}$ pasa a nivel BAJO se denomina *tiempo de establecimiento de dirección, $t_{S(A)}$.* El tiempo que la entrada $\overline{WE}$ debe estar a nivel BAJO es la anchura del impulso de escritura. El tiempo que la entrada $\overline{WE}$ debe permanecer a

nivel BAJO después de haber aplicado los datos válidos en las entradas de datos se designa por t_{WD}. El tiempo que los datos de entrada válidos deben permanecer en las líneas de entrada después de que la entrada $\overline{WE}$ pase a nivel ALTO es el *tiempo de mantenimiento de datos*, $t_{h(D)}$

En cada ciclo de escritura, se escribe en la memoria una unidad de datos.



(a) Ciclo de lectura ($\overline{WE}$ a nivel ALTO).



(b) Ciclo de escritura ($\overline{WE}$ a nivel BAJO)

FIGURA 10.12 Diagramas de tiempos de los ciclos de lectura y escritura básicos para la SRAM de la Figura 10.11.

SRAM síncrona de ráfaga

A diferencia de la SRAM asíncrona, una SRAM síncrona está sincronizada con el reloj del sistema. Por ejemplo, en un sistema informático, la SRAM síncrona opera con la misma señal de reloj que el microprocesador, de modo que el microprocesador y la memoria están sincronizados para conseguir una operación más rápida.

El concepto fundamental en que se basa la naturaleza síncrona de una SRAM puede ilustrarse con la Figura 10.13, que es un diagrama de bloques simplificado de una memoria de 32 k × 8, con propósitos ilustrativos. La SRAM síncrona es muy similar a la SRAM asíncrona, en términos de la matriz de memoria, del decodificador de direcciones y de las entradas de lectura/escritura y activación. La diferencia fundamental es que la SRAM síncrona utiliza registros con señal de reloj para sincronizar todas las entradas con el reloj del sistema. Tanto la dirección, como la entrada de lectura/escritura, la señal de activación del chip y los datos de

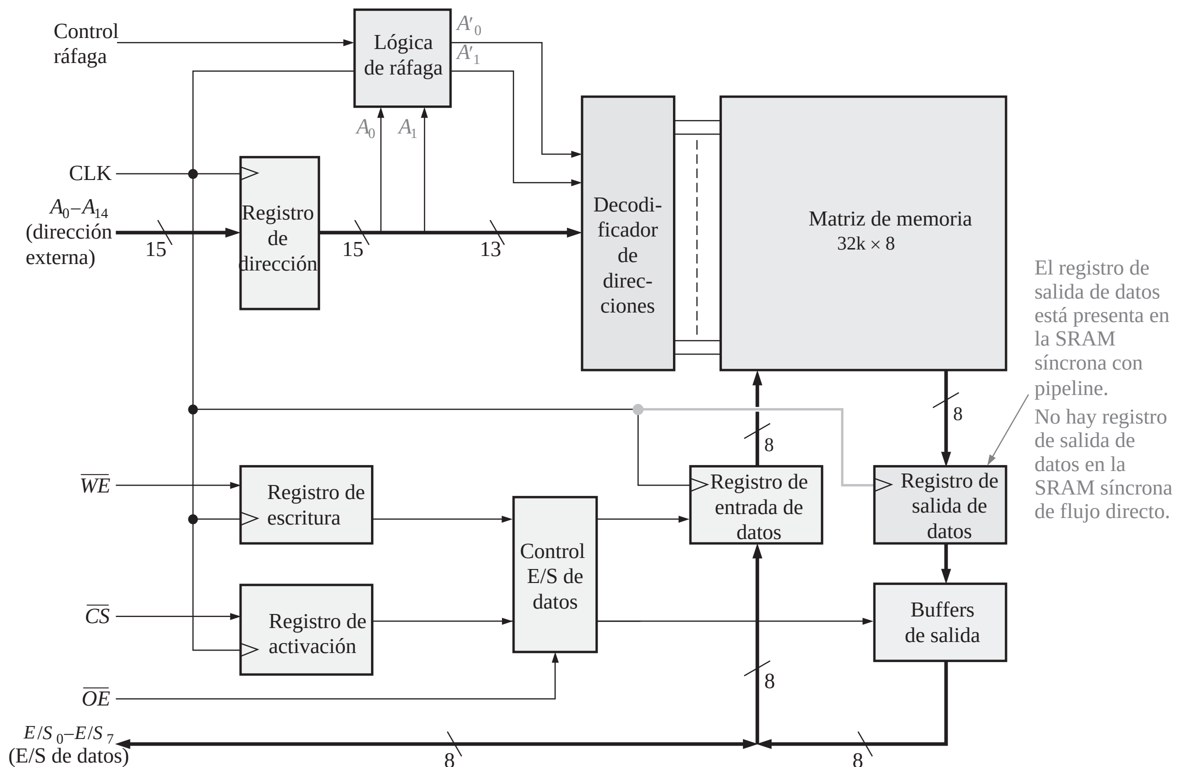
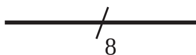


FIGURA 10.13 Diagrama de bloques básico de una SRAM de ráfaga síncrona.

entrada se enclavan en sus respectivos registros con el flanco activo del pulso de reloj. Una vez enclavada esta información, la operación de la memoria estará en sincronía con el reloj.

Para simplificar, se presenta en la Figura 10.13 una notación para expresar múltiples líneas paralelas o líneas de bus, como alternativa a dibujar cada línea por separado. Un conjunto de líneas paralelas puede expresarse mediante una sola línea gruesa atravesada por una barra y con una indicación del número de líneas distintas que forman el conjunto. Por ejemplo, la siguiente notación representa un conjunto de ocho líneas paralelas:



Los bits de dirección A_0 a A_{14} se enclavan en el registro de dirección con el flanco positivo de un pulso de reloj. En el mismo pulso de reloj, el estado de las líneas de activación de escritura $(\overline{WE})$ y de selección de chip $(\overline{CS})$ se enclava en el registro de escritura y el registro de activación, respectivamente. Éstos son registros de un único bit o simplemente flip-flops. Asimismo, los datos de entrada son enclavados con el mismo pulso de reloj en el registro de datos de entrada para las operaciones de escritura, y los datos existentes en una dirección de memoria seleccionada se enclavan en el registro de salida de datos para las operaciones de lectura, según determine el control de E/S de datos, basándose en las entradas procedentes del registro de escritura, del registro de activación y de la línea de activación de salida $(\overline{OE})$.

Existen dos tipos básicos de memoria SRAM síncronas: de *flujo directo* y con *pipeline*. La SRAM síncrona de flujo directo no dispone de un registro de salida de datos, por lo que los datos de salida fluyen asíncronamente hacia las líneas de E/S de datos a través de los búferes de salida. La SRAM síncrona con ***pipeline***

dispone de un registro de salida de datos, como se muestra en la Figura 10.13, por lo que los datos de salida se presentan síncronamente en las líneas de E/S de datos.

Operación en modo ráfaga. Como muestra la Figura 10.13, las memorias SRAM síncronas tienen normalmente una función de ráfaga de direcciones, que permite a la memoria leer o escribir en hasta cuatro posiciones utilizando una única dirección. Cuando se enclava una dirección externa en el registro de direcciones, los dos bits menos significativos de la dirección, A_0 y A_1, se aplican al circuito de la lógica de ráfaga. Éste produce una secuencia de cuatro direcciones internas añadiendo 00, 01, 10 y 11 a los dos bits de dirección menos significativos en sucesivos pulsos de reloj. La secuencia comienza siempre con la dirección base, que es la dirección externa almacenada en el registro de dirección.

La lógica de la ráfaga de direcciones en una SRAM síncrona típica está compuesta por un contador binario y puertas OR-exclusiva, como muestra la Figura 10.14. Para una lógica de ráfaga de 2 bits, la secuencia interna de ráfaga de direcciones se forma a partir de los bits A_2-A_{14} de la dirección base, más los dos bits de la dirección de ráfaga, A_1' y A_0'.

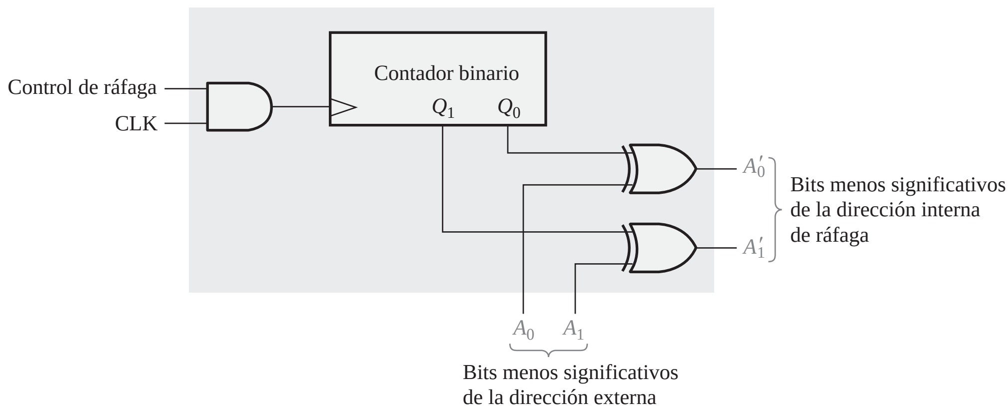


FIGURA 10.14 Lógica de la ráfaga de direcciones.

Al comenzar la secuencia de ráfaga, el contador se encuentra en su estado 00 y los dos bits menos significativos de la dirección se aplican a las entradas de las puertas XOR. Suponiendo que tanto A_0 como A_1 sean 0, los dos bits menos significativos de la secuencia de direcciones interna serían 00, 01, 10 y 11.

Memoria caché

Una de las principales aplicaciones de las memorias SRAM es la implementación de memorias caché en computadoras. La **memoria caché** es una memoria de alta velocidad y relativamente pequeña que almacena los datos o instrucciones más recientemente utilizados de la memoria principal, más grande pero más lenta. La memoria caché puede también utilizar memoria RAM dinámica (DRAM), de la que hablaremos a continuación. Normalmente, la memoria SRAM es varias veces más rápida que la memoria DRAM. En conjunto, la memoria caché hace que el microprocesador pueda acceder a la información almacenada mucho más rápido que si sólo se empleara memoria DRAM de alta capacidad. La memoria caché es, básicamente, un método eficiente en términos de coste para mejorar el rendimiento del sistema sin tener que incurrir en el gasto de hacer que toda la memoria sea más rápida.

El concepto de memoria caché se basa en la idea de que los programas informáticos tienden a obtener instrucciones o datos de un área de la memoria principal antes de pasar a otra área. Básicamente, el controlador de la caché "adivina" qué área de la lenta memoria dinámica necesitará a continuación la unidad central de

proceso (CPU), y mueve el contenido de dicha área a la memoria caché, para que esté listo cuando sea necesario. Si el controlador de caché ha realizado una estimación correcta, los datos están disponibles de manera inmediata para el microprocesador. Si la estimación del controlador de caché es errónea, la CPU debe acudir a la memoria principal y esperar mucho más tiempo para obtener las instrucciones o datos correctos. Afortunadamente, el controlador de caché tiene razón la mayor parte de las veces.

Analogía de la caché. Hay muchas analogías que pueden usarse para describir una memoria caché, pero tal vez lo más efectivo sea compararla con una nevera doméstica. Una nevera doméstica puede considerarse como una especie de "caché" para determinados productos alimenticios, mientras que el supermercado es la memoria principal donde se almacena toda la comida. Cada vez que deseamos comer o beber algo, podemos ir primero a la nevera (caché), para ver si contiene el producto que buscamos. Si es así, nos ahorramos un montón de tiempo. Si el producto no se encuentra allí, tendremos que invertir un tiempo adicional en obtenerlo del supermercado (memoria principal).

Cachés L1 y L2. Las cachés de nivel 1 (caché L1) están usualmente integradas en el chip del procesador y tienen una capacidad de almacenamiento muy limitada. La caché L1 se conoce también con el nombre de caché *primaria*. Una caché de nivel 2 (caché L2) es un chip o conjunto de chips de memoria independiente, externo al procesador, y usualmente dispone de una capacidad de almacenamiento mayor que una caché L1. La caché L2 también se conoce con el nombre de caché *secundaria*. Algunos sistemas pueden tener cachés de nivel superior (L3, L4, etc.), pero L1 y L2 son los más comunes. Asimismo, algunos sistemas emplean una caché de disco, ubicada en la memoria DRAM principal y utilizada para mejorar el rendimiento del disco duro, porque la DRAM, aunque mucho más lenta que la memoria SRAM, sigue siendo mucho más rápida que la unidad de disco duro. La Figura 10.15 ilustra dos memorias caché L1 y L2 en un sistema informático.

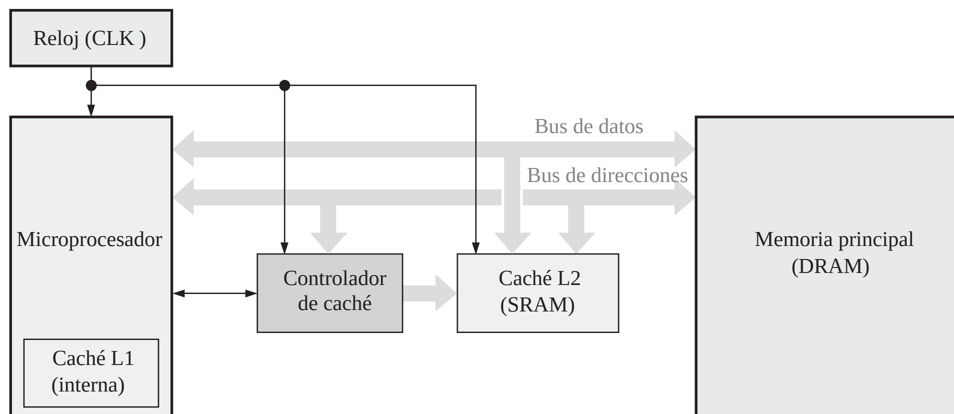


FIGURA 10.15 Diagrama de bloques mostrando memorias caché L1 y L2 en un sistema informático.

Las celdas de almacenamiento de la RAM dinámica (DRAM)

Las celdas de las **memorias dinámicas** almacenan un bit de datos en un condensador en lugar de en un *latch.* La ventaja de este tipo de celda es que es muy sencilla, lo que permite construir matrices de memoria muy grandes en un chip, a un coste por bit más bajo que el de las memorias estáticas. La desventaja es que el condensador de almacenamiento no puede mantenerse cargado más que un período de tiempo, y el dato almacenado se pierde a no ser que su carga se refresque periódicamente. La operación de refresco requiere circuitería de memoria adicional y complica el funcionamiento de la ***DRAM***. La Figura 10.16 presenta una celda típica de una DRAM, formada por un único transistor MOS (MOSFET) y un condensador.

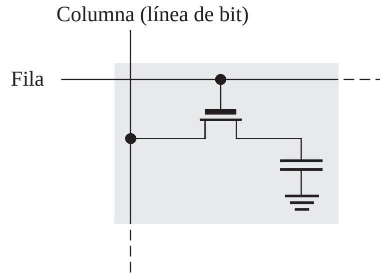


FIGURA 10.16 Celda de una RAM dinámica MOS.

En este tipo de celda, el transistor actúa como un interruptor. El funcionamiento básico simplificado se ilustra en la Figura 10.17, y es el siguiente. Un nivel BAJO en la línea $R/\overline{W}$ (modo escritura) activa el buffer de tres estados de entrada y desactiva el *buffer* de salida. Para escribir un 1 en la celda, la línea D_{IN} debe estar a nivel ALTO, y el transistor debe ser puesto en conducción poniendo un nivel ALTO en la línea de fila. El transistor opera como un interruptor cerrado, que conecta el condensador a la línea de bit. Esta conexión permite al condensador cargarse con una tensión positiva, como muestra la Figura 10.17(a). Cuando se almacena un 0, se aplica un nivel BAJO a la línea D_{IN}. Si el condensador almacenaba un 0, permanece descargado; ahora bien, si almacenaba un 1, se descarga como se indica en la Figura 10.17(b). Cuando la línea de fila vuelve al nivel BAJO, el transistor no conduce y desconecta el condensador de la línea de bit, con lo que la carga (1 o 0) "queda atrapada" en el condensador.

Para leer una celda, la línea $R/\overline{W}$ (*Read*/$\overline{\textit{Write}}$, lectura/$\overline{\text{escritura}}$) se pone a nivel ALTO, lo que activa el buffer de salida y desactiva el *buffer* de entrada. Cuando la línea de fila se pone a nivel ALTO, el transistor conduce y conecta el condensador a la línea de bit y, por tanto, al buffer de salida (amplificador). De esta manera, el bit de datos aparece en la línea de salida de datos (D_{OUT}). Este proceso se ilustra en la Figura 10.17(c).

Para refrescar la celda de memoria, la línea $R/\overline{W}$, la línea de fila y la línea de refresco se ponen a nivel ALTO. El transistor conduce, conectando el condensador a la línea de bit. El *buffer* de salida se activa y el bit de datos almacenado se aplica a la entrada del *buffer* de refresco, el cual se activa mediante el nivel ALTO de la entrada de refresco. Esto da lugar a una tensión en la línea de bit que corresponde al bit almacenado, recargando el condensador como se ilustra en la Figura 10.17(d).

Organización básica de una DRAM

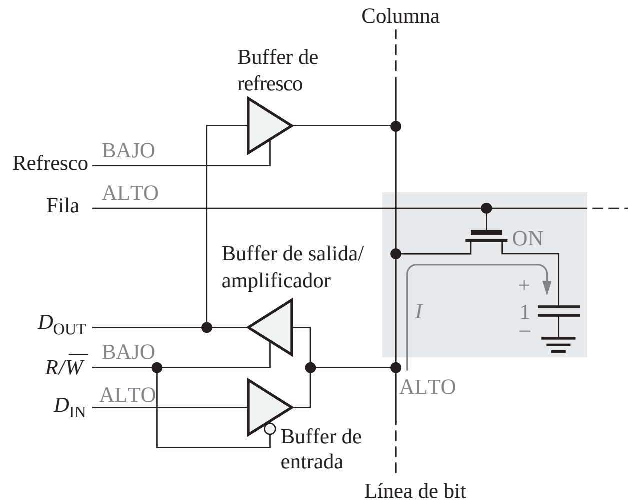
La principal aplicación de las DRAM se encuentra en la memoria principal de las computadoras. La diferencia principal entre las DRAM y las SRAM es el tipo de celda de memoria. Como se ha visto, la celda de la memoria DRAM está formada por un transistor y un condensador, y es mucho más sencilla que la celda de la SRAM. Esto permite densidades mucho mayores en las DRAM, lo que da lugar a mayores capacidades de bits para una determinada área de chip, aunque el tiempo de acceso es mucho mayor.

De nuevo, dado que la carga almacenada en un condensador tiende a perderse, las celdas de una DRAM requieren una operación de refresco frecuente para conservar los bits de datos almacenados. Este requisito da lugar a una circuitería más compleja que en la SRAM. A continuación se exponen varias funciones comunes en la mayor parte de las DRAM, utilizando como ejemplo una DRAM genérica de 1 M × 1 bit.

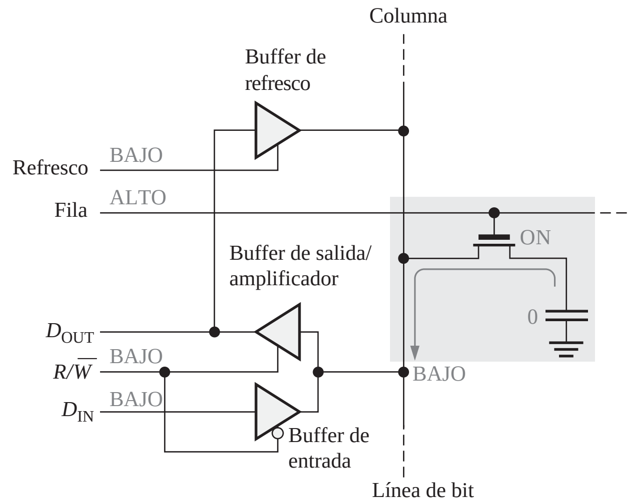
Multiplexación de direcciones. Las DRAM utilizan una técnica denominada *multiplexación de direcciones* que sirve para reducir el número de líneas de dirección. La Figura 10.18 muestra el diagrama de bloques de una DRAM de 1.048.576 bits (1 Mbit) con una organización de 1M × 1. Vamos a centrarnos en los bloques de color gris oscuro para ilustrar la multiplexación de direcciones. Los bloques en color gris claro representan la lógica de refresco.

Las diez líneas de dirección se multiplexan en el tiempo al comienzo de un ciclo de memoria mediante la validación de dirección de fila ($\overline{RAS}$) y la validación de dirección de columna ($\overline{CAS}$), en dos campos de dirección separados. En primer lugar, la dirección de fila de 10 bits se pasa al *latch* de direcciones de fila. Después, la dirección de columna de 10 bits se pasa al *latch* de direcciones de columna. Las direcciones de fila y columna se decodifican para seleccionar una de las 1.048.576 (2^{20} = 1.048.576) direcciones de la matriz de memoria. En la Figura 10.19 se presenta el diagrama de tiempos básico para la operación de multiplexación de direcciones.

Ciclos de lectura y escritura. Al inicio de cada ciclo de memoria de lectura o escritura, $\overline{RAS}$ y $\overline{CAS}$ se activan (nivel BAJO) para multiplexar las direcciones de fila y columna hacia los *latches* y decodificadores. Durante

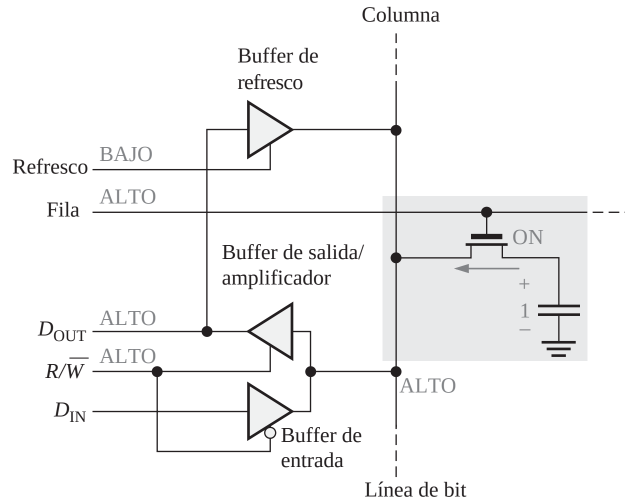


(a) Escritura de un 1 en la celda de memoria

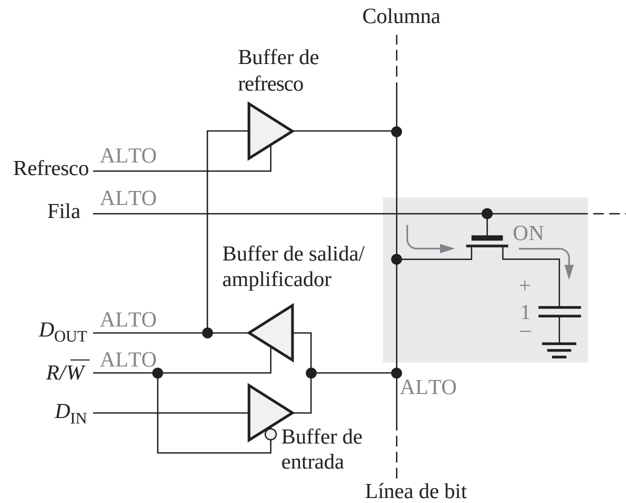


(b) Escritura de un 0 en la celda de memoria

FIGURA 10.17 Funcionamiento básico de una celda de una DRAM. (*Cóntinúa*)



(c) Lectura de un 1 desde la celda de memoria



(d) Refresco de un 1 almacenado

FIGURA 10.17 Funcionamiento básico de una celda de una DRAM. (*Cóntinuación*)

el ciclo de lectura, la entrada $R/\overline{W}$ está a nivel ALTO. Durante el ciclo de escritura, la entrada $R/\overline{W}$ está a nivel BAJO. Esto se ilustra en la Figura 10.20.

Modo página rápido. En los ciclos de lectura y escritura normales descritos anteriormente, primero se carga la dirección de fila de una posición de memoria concreta mediante la entrada activa a nivel BAJO $\overline{RAS}$, y luego se carga la dirección de columna de esa posición mediante la entrada activa a nivel BAJO $\overline{CAS}$. Después se selecciona la siguiente posición mediante otra entrada $\overline{RAS}$ seguida de $\overline{CAS}$, y así sucesivamente.

Una "página" es una sección de memoria disponible en una misma dirección de fila y que consta de todas las columnas de dicha fila. El modo página rápido permite operaciones de lectura y escritura sucesivas en cada una de las direcciones de columna de una fila seleccionada. En primer lugar, se carga una dirección de fila

FIGURA 10.18 Diagrama de bloques simplificado de una DRAM de 1M × 1.

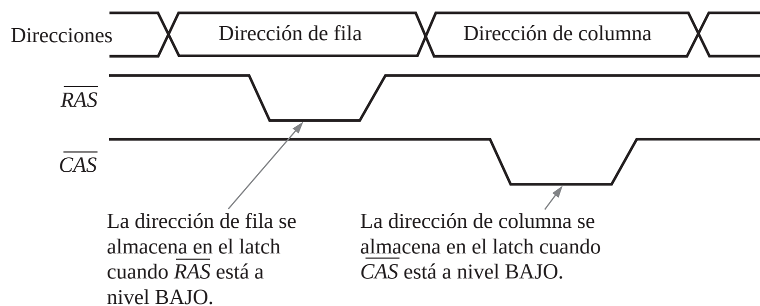
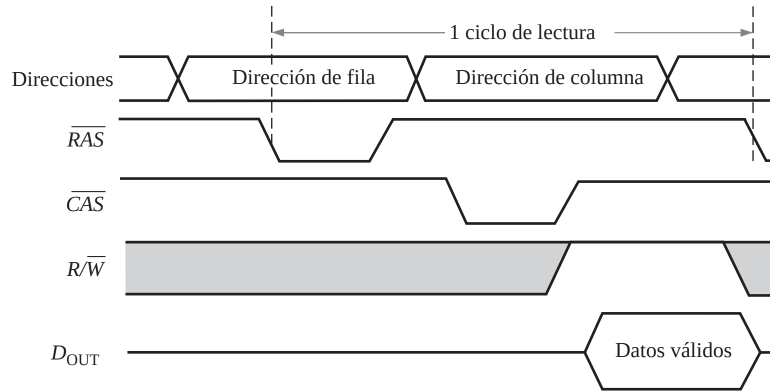
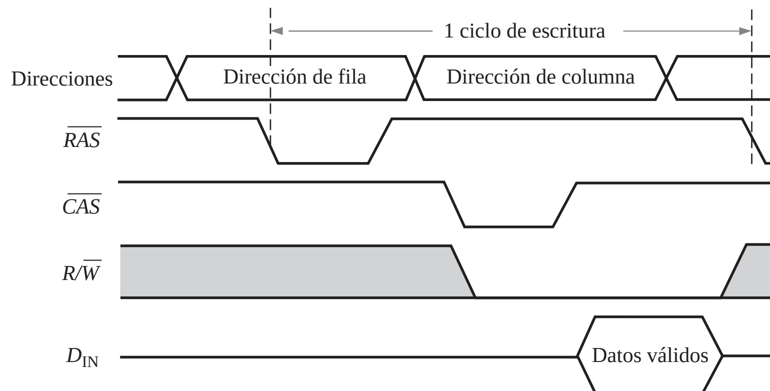


FIGURA 10.19 Diagrama de tiempos básico para la multiplexación de direcciones.

pasando la entrada $\overline{RAS}$ a nivel BAJO y permaneciendo en este nivel, mientras que $\overline{CAS}$ bascula entre los niveles ALTO y BAJO. Sólo se selecciona una dirección de fila, que permanece seleccionada mientras que $\overline{RAS}$ esté activa. Cada $\overline{CAS}$ sucesiva selecciona una columna de la fila especificada. De este modo, después



(a) Ciclo de lectura



(b) Ciclo de escritura

FIGURA 10.20 Diagrama de tiempos para los ciclos de lectura y escritura normales.

de un ciclo en modo página rápido, todas las direcciones de la fila seleccionada se habrán leído o escrito, dependiendo del nivel de $R/\overline{W}$. Por ejemplo, un ciclo en modo página rápido para la DRAM de la Figura 10.18 requiere que la señal $\overline{CAS}$ se active 1024 veces para cada fila seleccionada mediante la señal $\overline{RAS}$.

En la Figura 10.21 se presenta un diagrama de tiempos que ilustra el funcionamiento básico en modo página rápida para la operación de lectura. Cuando $\overline{CAS}$ pasa a su estado no activo (ALTO), desactiva las salidas de datos. Por tanto, la transición de la señal $\overline{CAS}$ al nivel ALTO sólo debe producirse después de que el sistema externo almacene los datos válidos en un *latch*.

Ciclos de refresco. Como ya sabe, las DRAM se basan en el almacenamiento de carga en un condensador para cada bit de memoria de la matriz. Esta carga se degrada (se pierde) con el tiempo y la temperatura, por lo que cada bit se debe refrescar (recargar) periódicamente para mantener el estado correcto del bit. Típicamente, una DRAM se debe refrescar cada 8 ms o 16 ms, aunque en algunos dispositivos el período de refresco puede exceder 100 ms.

Una operación de lectura refresca automáticamente todas las direcciones de la fila seleccionada. Sin embargo, en aplicaciones típicas, no siempre se puede predecir cuán a menudo se producirá un ciclo de lectura y, por tanto, no se puede depender de que un ciclo de lectura se efectúe frecuentemente para evitar la pérdida de datos. En consecuencia, en los sistemas DRAM se deben implementar ciclos de refresco especiales.

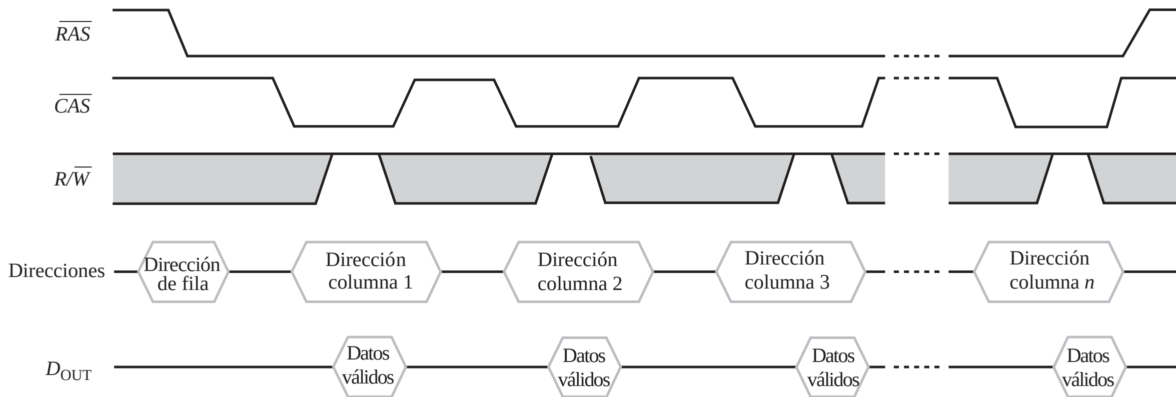


FIGURA 10.21 Cronograma del modo página rápido para la operación de lectura.

El *refresco a ráfagas* y el *refresco distribuido* son los dos modos básicos de las operaciones de refresco. En el refresco a ráfagas, todas las filas de la matriz de memoria se recargan consecutivamente en cada período de refresco. Para una memoria con un período de refresco de 8 ms, se produce una vez un refresco a ráfagas de todas las filas cada 8 ms. Las operaciones de lectura y escritura normales se suspenden durante un ciclo de refresco de ráfaga. En el refresco distribuido, cada fila se refresca en intervalos entremezclados con los ciclos de lectura y escritura normales. Por ejemplo, en la memoria de la Figura 10.18, que tiene 1024 filas, un periodo de refresco de 8 ms exige que se recargue una fila cada 8ms/1024 = 7,8 μs, cuando se utiliza el refresco distribuido.

Los dos tipos de operaciones de refresco son: *refresco sólo con* $\overline{RAS}$ ($\overline{RAS}$*-only refresh*) y *refresco* $\overline{CAS}$ *antes de* $\overline{RAS}$ ($\overline{CAS}$ *before* $\overline{RAS}$ *refresh*). El refresco sólo $\overline{RAS}$ consiste en una transición de la señal $\overline{RAS}$ a nivel BAJO (estado activo), que almacena la dirección de la fila en el *latch* para realizar el refresco, mientras que la línea $\overline{CAS}$ permanece a nivel ALTO (estado inactivo) a lo largo del ciclo. Se utiliza un contador externo para proporcionar las direcciones de fila en este tipo de operación.

El refresco $\overline{CAS}$ antes de $\overline{RAS}$ se inicia cuando la línea $\overline{CAS}$ pasa a nivel BAJO y a continuación la línea $\overline{RAS}$ pasa a nivel BAJO. Esta secuencia activa un contador de refresco interno que genera la dirección de fila para la que se debe realizar la recarga. Esta dirección se conmuta mediante el selector de datos hacia el decodificador de filas.

Tipos de memoria DRAM

Ahora que hemos aprendido los conceptos básicos de una memoria DRAM, echemos un breve vistazo a los tipos principales existentes. Dichos tipos son la *DRAM con modo página rápido (Fast Page Mode, FPM DRAM)*, la *DRAM con salida de datos extendida (Extended Data Output, EDO DRAM)*, la *DRAM con salida de datos extendida en ráfaga (Burst Extended Data Output, BEDO DRAM)* y la *DRAM síncrona (Synchronous DRAM, SDRAM)*.

FPM DRAM. Ya hemos descrito anteriormente la operación en modo de página rápido. Este tipo de DRAM ha sido tradicionalmente el más común y es el que se ha usado en las computadoras hasta el desarrollo de la EDO DRAM. Recuerde que una página de la memoria está formada por todas las direcciones de columna contenidas en una misma dirección de fila.

Como ya hemos visto, la idea básica de la **FPM DRAM** se basa en la probabilidad de que las siguientes direcciones de memoria a las que haya que acceder se encuentren en la misma fila (en la misma página). Afortunadamente, esto sucede en un gran porcentaje de las veces. El modo FPM ahorra tiempo, con respecto

al acceso puramente aleatorio, porque en el modo FPM la dirección de fila se especifica una única vez para acceder a varias direcciones de columna sucesivas, mientras que en el acceso aleatorio puro, hay que especificar una dirección de fila para cada dirección de columna.

Recuerde que en una operación de lectura en modo de página rápido, la señal $\overline{CAS}$ tiene que esperar hasta que los datos válidos correspondientes a una dirección dada sean aceptados (enclavados) por el sistema externo (UCP), antes de volver a su estado inactivo. Cuando $\overline{CAS}$ pasa a su estado inactivo, se desactivan las salidas de datos. Esto significa que la siguiente dirección de columna no puede ser generada hasta que los datos correspondientes a la dirección de columna actual sean transferidos a la UCP. Esto limita la velocidad de acceso a las columnas situadas dentro de una página.

EDO DRAM. La memoria DRAM con salida de datos extendida, algunas veces denominada *DRAM con modo hiperpágina*, es muy similar a la FPM DRAM. La diferencia fundamental es que la señal $\overline{CAS}$ en la **EDO DRAM.** no desactiva los datos de salida cuando pasa a su estado de inactividad, porque se pueden mantener los datos válidos correspondientes a la dirección actual hasta que $\overline{CAS}$ vuelva a activarse. Esto significa que se puede acceder a la siguiente dirección de columna antes de que el sistema externo acepte los datos válidos actuales. La idea es acelerar el tiempo de acceso.

BEDO DRAM. La DRAM con salida de datos extendida en ráfaga es una EDO DRAM con la capacidad de generar ráfagas de direcciones. Recuerde, de nuestra explicación sobre la SRAM síncrona de ráfaga, que la función de ráfaga de direcciones permite generar internamente hasta cuatro direcciones a partir de una única dirección externa, lo que ahorra tiempo de acceso. Este mismo concepto se aplica a la **BEDO DRAM**.

SDRAM. Para poder estar a la altura de la siempre creciente velocidad de los microprocesadores, son necesarias memorias DRAM más rápidas. La DRAM síncrona es uno de los esfuerzos más recientes en este sentido. Al igual que la RAM estática síncrona explicada anteriormente, la operación de la memoria **SDRAM** está sincronizada con el reloj del sistema, con el que también opera el microprocesador de un sistema informático. Las mismas ideas básicas descritas en relación con la SRAM síncrona de ráfaga se pueden aplicar a la memoria SDRAM.

Esta operación de tipo síncrono hace que la memoria SDRAM sea totalmente diferente de los otros tipos de DRAM asíncrona previamente mencionados. Con las memorias asíncronas, el microprocesador se ve obligado a esperar a que la DRAM complete sus operaciones internas. Sin embargo, con la operación de tipo síncrono, la DRAM enclava las direcciones, los datos y la información de control generados por el procesador, bajo control del reloj del sistema. Esto permite al procesador gestionar otras tareas mientras se están realizando las operaciones de lectura o escritura en memoria, en lugar de tener que esperar a que la memoria realice su tarea, como es el caso en los sistemas asíncronos.

REVISIÓN DE LA SECCIÓN 10.2

1. Enumerar dos tipos de SRAM.
2. ¿Qué es una memoria caché?
3. Explicar en qué se diferencian las SRAM y las DRAM.
4. Describir la operación de refresco de una DRAM.
5. Enumerar cuatro tipos de DRAM.

10.3 MEMORIAS DE SÓLO LECTURA (ROM)

Una ROM mantiene de forma permanente o semipermanente los datos almacenados, que pueden ser leídos de la memoria pero, o no se pueden cambiar en absoluto, o se requiere un equipo especial para ello. Una ROM almacena datos que se utilizan repetidamente en las aplicaciones, tales como tablas,

conversiones o instrucciones programadas para la inicialización y el funcionamiento de un sistema. Las ROM mantienen los datos almacenados cuando se desconecta la alimentación y son, por tanto, memorias no volátiles.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Enumerar los tipos de ROM. ■ Describir una celda básica de almacenamiento ROM de máscara. ■ Explicar cómo se leen los datos de una ROM. ■ Estudiar la organización interna de una ROM típica. ■ Estudiar algunas aplicaciones de las ROM.

La familia de las memorias ROM

La Figura 10.22 muestra cómo se clasifican las memorias ROM semiconductoras. La ROM de máscara es un tipo de memoria en la que los datos se almacenan permanentemente en la memoria durante el proceso de fabricación. La **PROM**, o ROM programable, es aquel tipo de ROM en la que el usuario, con ayuda de equipos especializados, almacena eléctricamente los datos. Tanto la ROM de máscara como la PROM pueden ser de cualquier tecnología MOS o bipolar. La **EPROM**, o memoria PROM borrable (*erasable* PROM) es exclusivamente un dispositivo MOS. La **UV EPROM** puede ser programada eléctricamente por el usuario, pero los datos almacenados deben borrarse mediante la exposición a la luz ultravioleta durante un período de varios minutos. La PROM borrable eléctricamente (**EEPROM** o E^2PROM, *Electrically Erasable* PROM) se puede borrar en unos pocos milisegundos.

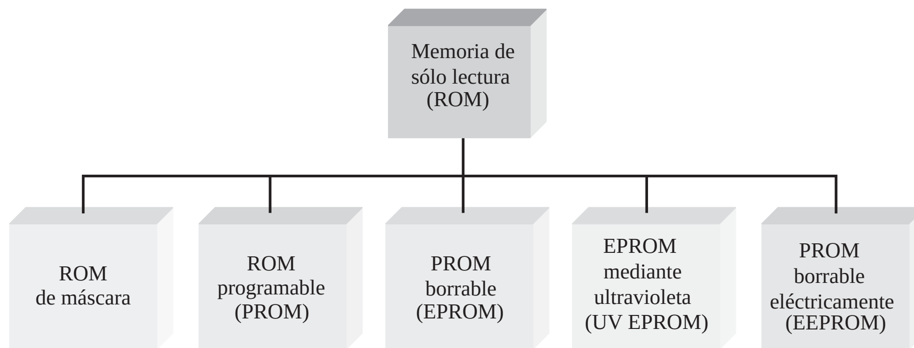


FIGURA 10.22 La familia de memorias ROM.

La ROM de máscara

Normalmente, la ROM de máscara se denomina simplemente ROM. Es una memoria programada de forma permanente durante el proceso de fabricación, para proporcionar funciones estándar de uso extendido, tales como conversiones populares, o para proporcionar funciones especificadas por el usuario. Una vez que se programa la memoria, ésta no puede cambiarse. La mayoría de los circuitos integrados ROM utilizan la presencia o ausencia de una conexión de transistor en una unión fila/columna para representar un 1 o un 0.

La Figura 10.23 muestra celdas MOS de una ROM. La presencia de una conexión desde una línea de fila a la puerta de un transistor representa un 1 en esa posición, ya que, cuando la línea de fila está a nivel ALTO, todos los transistores con conexión de puerta a esa línea de fila conducen, y ponen a nivel ALTO (1) a las líneas de columna asociadas. En las uniones de fila/columna en las que no existe conexión de puerta, las líneas de columna permanecen a nivel BAJO (0) cuando se direcciona la fila.

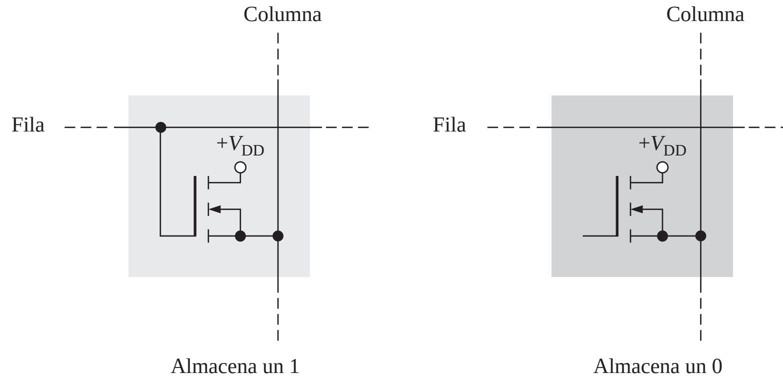


FIGURA 10.23 Celdas ROM.

Una ROM básica

Para ilustrar el concepto de ROM, la Figura 10.24 muestra una matriz ROM simplificada. Los cuadrados en color gris claro representan los 1s almacenados y los cuadrados en gris oscuro representan los 0s almacenados. El procedimiento de lectura básico es el siguiente: cuando se aplica un código de dirección binario a las líneas de entrada de dirección, la línea de la fila correspondiente se pone a nivel ALTO. Este nivel ALTO se conecta a las líneas de las columnas a través de los transistores en cada unión (celda) donde se almacena un 1. En cada celda en la que se almacena un 0, la línea de columna permanece a nivel BAJO, debido a la

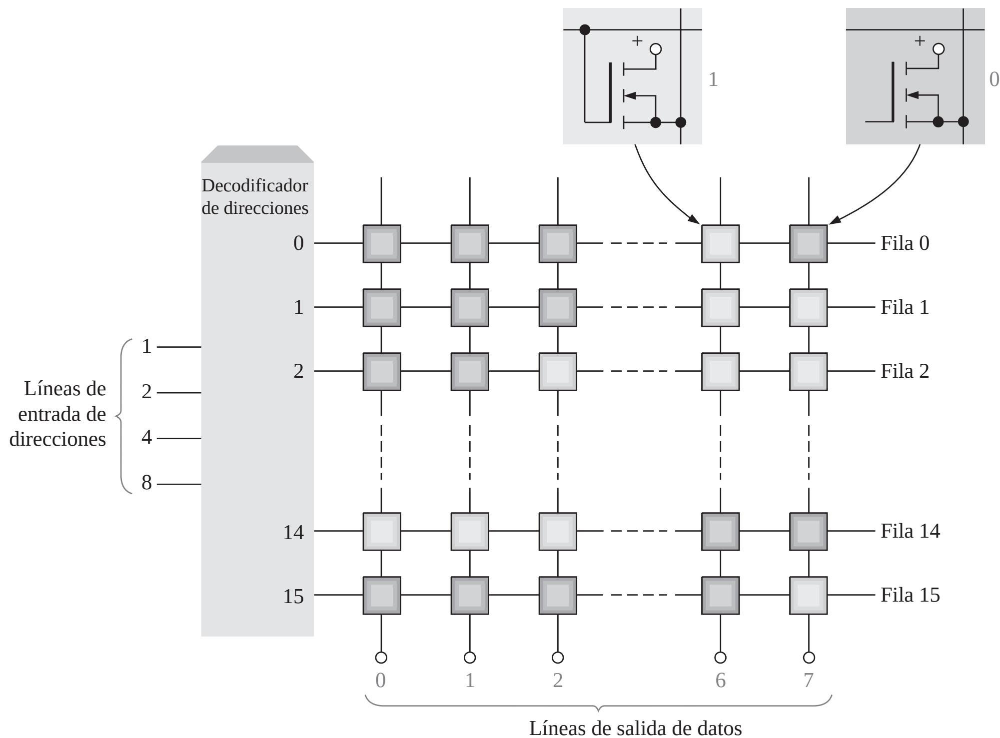


FIGURA 10.24 Representación de una matriz ROM de 16 × 8 bits.

resistencia de terminación. Las líneas de las columnas constituyen la salida de datos. Los ocho bits de datos almacenados en la fila seleccionada se presentan en las líneas de salida.

Como puede ver, la ROM de ejemplo de la Figura 10.24 está organizada en 16 direcciones, cada una de las cuales almacena 8 bits de datos. Por tanto, se trata de una ROM de 16 × 8 (16 por 8) y su capacidad total es de 128 bits, o 16 bytes. Las ROM puede utilizar tablas de búsqueda (LUT, *Look-Up Table*) para realizar conversiones de códigos y generación de función lógicas.

EJEMPLO 10.1

Dibujar una ROM similar a la de la Figura 10.24, programada para convertir a código Gray números binarios de 4 bits.

Solución Repase, en el Capítulo 2, el código Gray. Se ha desarrollado la Tabla 10.1 con el fin de utilizarla para programar la ROM.

Binario				Gray			
B_3	B_2	B_1	B_0	G_3	G_2	G_1	G_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

TABLA 10.1

La matriz ROM resultante de 16 × 4 se muestra en la Figura 10.25. Se puede ver que un código binario en las líneas de dirección de entrada produce el correspondiente código Gray en las líneas de salida (columnas). Por ejemplo, cuando se aplica el número binario 0110 a las líneas de dirección de entrada, se selecciona la dirección 6, que almacena el código Gray 0101.

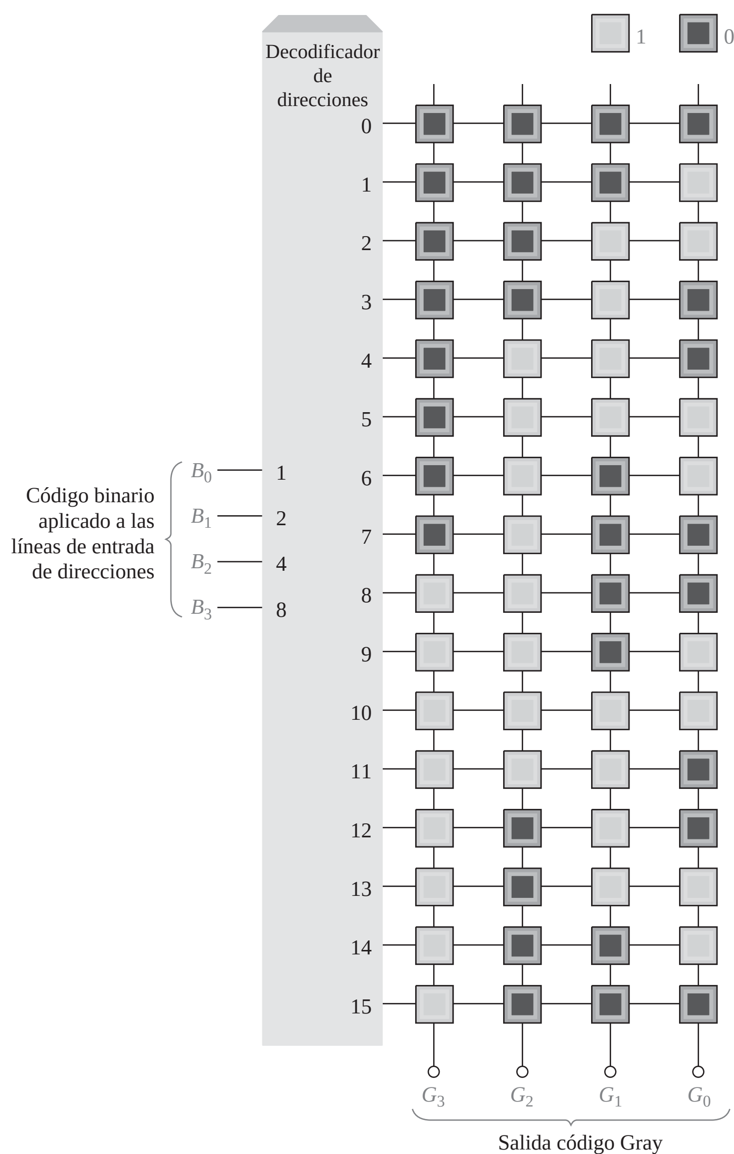


FIGURA 10.25 Representación de una ROM programada como convertidor de código binario - Gray.

Problema relacionado* Utilizando la Figura 10.25, determinar el código Gray de salida cuando se aplica el código binario 1011 a las líneas de dirección de entrada.

*Las respuestas se encuentran al final del capítulo.

Organización interna de la ROM

La mayoría de los circuitos integrados ROM tienen una organización interna algo más compleja que la de la ROM básica del ejemplo que acabamos de presentar. Para ilustrar cómo se estructura un CI ROM se utiliza

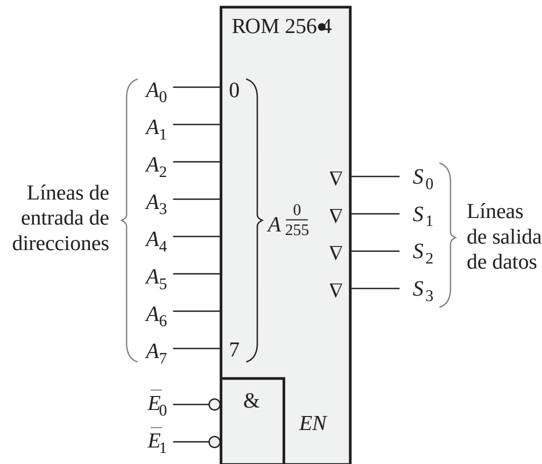


FIGURA 10.26 Símbolo lógico de una ROM de 256 × 4. El identificativo $A\frac{0}{255}$ significa que el código de dirección de 8 bits selecciona direcciones de 0 a 255.

un dispositivo de 1024 bits organizado en una matriz de 256 × 4. El símbolo lógico se muestra en la Figura 10.26. Cuando se aplica cualquiera de los 256 códigos binarios (ocho bits) a las líneas de dirección, se presentan cuatro bits de datos en las salidas si las entradas de habilitación del chip se encuentran a nivel BAJO (256 direcciones requieren ocho líneas de dirección).

Aunque la estructura de 256 × 4 de este dispositivo parece indicar que tiene 256 filas y 4 columnas en la matriz de memoria, éste no es el caso en realidad. Realmente la matriz de memoria es una matriz de 32 × 32 (32 filas y 32 columnas), como muestra el diagrama de bloques de la Figura 10.27

La ROM de la Figura 10.27 funciona de la manera siguiente: cinco de las ocho líneas de dirección (A_0 hasta A_4) se decodifican mediante el decodificador de filas (comúnmente denominado decodificador *Y*) para seleccionar una de las 32 filas. Tres de las ocho líneas de dirección (A_5 hasta A_7) se decodifican mediante el decodificador de columnas (denominado comúnmente decodificador *X*) para seleccionar cuatro de las 32 columnas. En realidad, el decodificador de columnas está formado por cuatro decodificadores 1-de-8 (selectores de datos), como se muestra en la Figura 10.27.

El resultado de esta estructura es que, al aplicar un código de dirección de 8 bits (A_0 hasta A_7), aparece una palabra de datos de 4 bits en las salidas de datos cuando las líneas de habilitación del chip ($\overline{E}_0$ y $\overline{E}_1$) deben estar a nivel BAJO para activar los buffers de salida. Este tipo de organización interna (arquitectura) es típica de diversos circuitos integrados ROM, de distintas capacidades.



NOTAS INFORMÁTICAS

La memoria ROM se usa en las computadoras personales para almacenar lo que se denomina BIOS (*Basic Input/Output Services*, servicios básicos de entrada/salida). Se trata de programas que se emplean para llevar a cabo funciones fundamentales de soporte y supervisión en la computadora. Por ejemplo, los programas de BIOS almacenados en la ROM controlan determinadas funciones de vídeo, proporcionan la función de formateo de discos, exploran el teclado para detectar las pulsaciones y controlan ciertas funciones de impresión.

Tiempo de acceso de la ROM

En la Figura 10.28 se presenta un diagrama de tiempos típico que ilustra el tiempo de acceso a la ROM. El **tiempo de acceso** de una ROM, t_a, es el tiempo que transcurre desde que se aplica un código de dirección váli

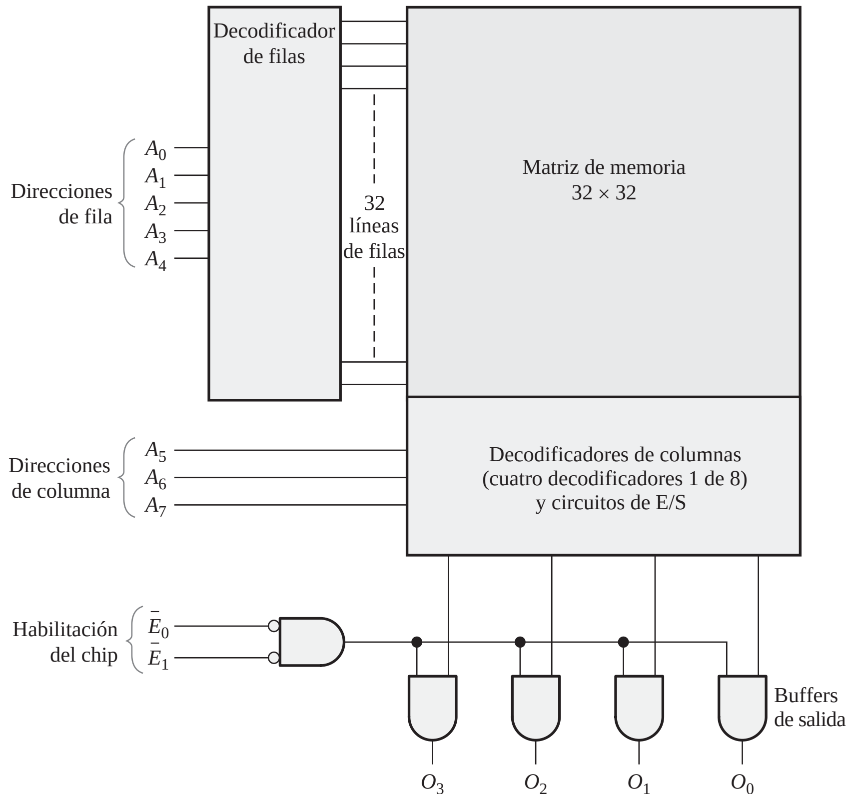


FIGURA 10.27 ROM de 1024 bits con una organización de 256 × 4 basada en una matriz de 32 × 32.

do en las líneas de entrada hasta que aparecen los datos válidos en las líneas de salida. El tiempo de acceso se puede también medir desde que se activa la entrada de habilitación del chip $(\bar{E})$ hasta que aparecen los datos válidos en la salida, cuando ya se encuentra una dirección válida en las líneas de entrada.

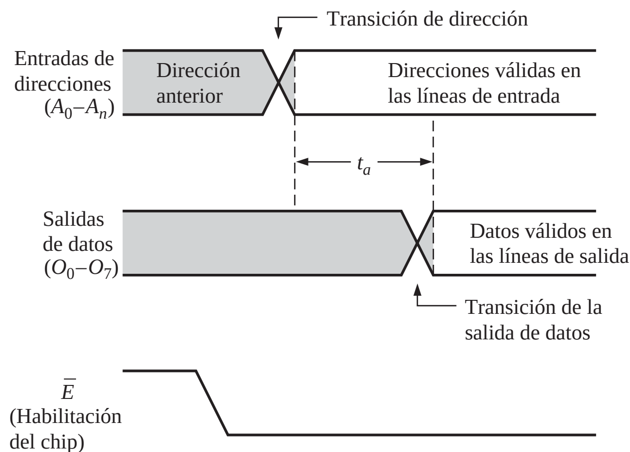


FIGURA 10.28 Tiempo de acceso de una ROM (t_a) desde el cambio de dirección hasta la salida de datos, con la entrada de habilitación del chip activa.

REVISIÓN DE LA SECCIÓN 10.3

1. ¿Cuál es la capacidad de almacenamiento en bits de una ROM con una estructura de 512×8?
2. Enumerar los tipos de las memorias de sólo lectura.
3. ¿Cuántos bits de dirección son necesarios en una memoria de 2048 bits organizada como una memoria de 256×8?

10.4 MEMORIAS ROM PROGRAMABLES (PROM Y EPROM)

Las PROM son básicamente iguales que las ROM de máscara, una vez que han sido programadas. Como ya hemos visto, las ROM son un tipo de dispositivo lógico programable. La diferencia consiste en que las PROM salen de fábrica sin estar programadas y se programan a medida para satisfacer las necesidades del usuario.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Distinguir entre una ROM de máscara y una PROM. ■ Describir una celda básica de almacenamiento de una PROM. ■ Utilizar las EPROM, incluyendo las UV EPROM y EEPROM. ■ Analizar un ciclo de programación de una EPROM.

Memorias PROM

Las ***PROM*** utilizan algún tipo de mecanismo de fundición para almacenar bits, donde un *hilo* de memoria se funde o queda intacto para representar un 0 o un 1. El proceso de fundición es irreversible; una vez que una PROM ha sido programada no puede cambiarse.

La Figura 10.29 muestra una matriz MOS de una PROM con hilos fusibles. Los fusibles se introducen en la PROM (durante el proceso de fabricación) entre la fuente del transistor de cada celda y su línea de columna. Durante el proceso de programación, se introduce una corriente adecuada a través del hilo fusible para fundirlo y que permanezca abierto, almacenando de esta manera un 0. El fusible se deja intacto para almacenar un 1.

Los tres tipos básicos de tecnologías de fusibles utilizados en las PROM son las conexiones de metal, las conexiones de silicio y las uniones *pn*. A continuación, se proporciona una breve descripción de cada una de ellas.

1. Las conexiones de metal se realizan con materiales como el nicromo. Cada bit de la matriz de memoria se representa mediante una conexión separada. Durante la programación, la conexión puede fundirse o quedar intacta. Básicamente, esto se realiza direccionando primero una determinada celda, y luego aplicando una cantidad de corriente suficientemente alta como para hacer que la conexión se abra.
2. Las conexiones de silicio están constituidas por tiras estrechas y alargadas de silicio policristalino. La programación de estos fusibles requiere que las conexiones se fundan por el paso de una cantidad de corriente adecuada a su través. Esta cantidad de corriente hace que aumente la temperatura en el fusible, lo que origina que se oxide el silicio, formando un aislante alrededor de la conexión que ahora está abierta.
3. La tecnología de uniones cortocircuitadas, o de migración inducida por avalancha, consiste básicamente en dos uniones *pn* dispuestas una frente a la otra. Durante el proceso de programación, una de las uniones de los diodos entra en avalancha, y el voltaje y el calor resultantes hacen que los iones de aluminio migren y cortocircuiten la unión. La unión restante se utiliza posteriormente como diodo polarizado en directa para representar un bit de datos.

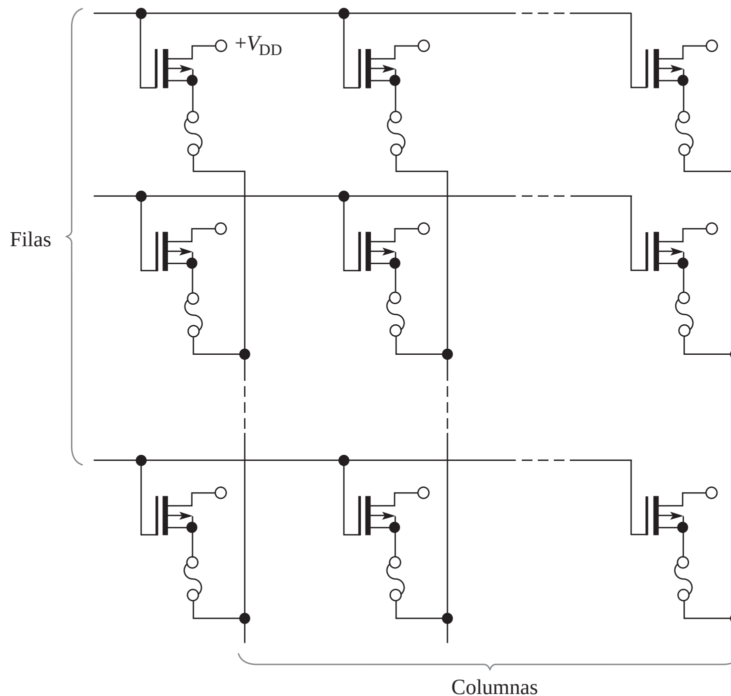


FIGURA 10.29 Matriz PROM MOS con hilos fusibles. Generalmente, todos los drenadores se conectan a V_{DD}.

Memorias EPROM

Una ***EPROM*** es una PROM borrable. A diferencia de una PROM ordinaria, una EPROM puede ser reprogramada si antes se borra el programa existente en la matriz de memoria.

Una EPROM utiliza una matriz NMOSFET con una estructura de puerta aislada. La puerta del transistor aislada no tiene ninguna conexión eléctrica y puede almacenar una carga eléctrica durante un período de tiempo indefinido. Los bits de datos en este tipo de matriz se representan mediante la presencia o ausencia de una carga almacenada en la puerta. El borrado de un bit de datos es un proceso que elimina la carga de la puerta.

Los dos tipos fundamentales de memorias PROM borrables son las PROM borrables por rayos ultravioleta (UV EPROM) y las PROM borrables eléctricamente (EEPROM).

UV EPROM. Una UV EPROM se puede reconocer por la ventana de cuarzo transparente de su encapsulado, como se muestra en la Figura 10.30. La puerta aislada del **FET** de una EPROM ultravioleta está "flotando" dentro de un material óxido aislante. El proceso de programación hace que los electrones sean

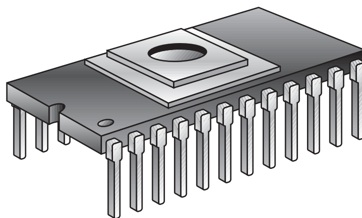


FIGURA 10.30 Encapsulado de una PROM borrable por rayos ultravioleta.

eliminados de la puerta flotante. El borrado se realiza mediante la exposición del chip de la matriz de memoria a una radiación ultravioleta de alta intensidad, a través de la ventana de cuarzo en la parte superior del encapsulado. La carga positiva almacenada en la puerta se neutraliza después de un período de tiempo de entre unos minutos y una hora de exposición.

En la Figura 10.31 se representa una UV EPROM típica mediante un diagrama lógico. Su funcionamiento es representativo de otras memorias UV EPROM típicas, de distintos tamaños. Como muestra el símbolo lógico de la Figura 10.31, este dispositivo tiene 2048 direcciones ($2^{11} = 2048$), cada una con ocho bits. Observe que las ocho salidas son tri-estado (∇).

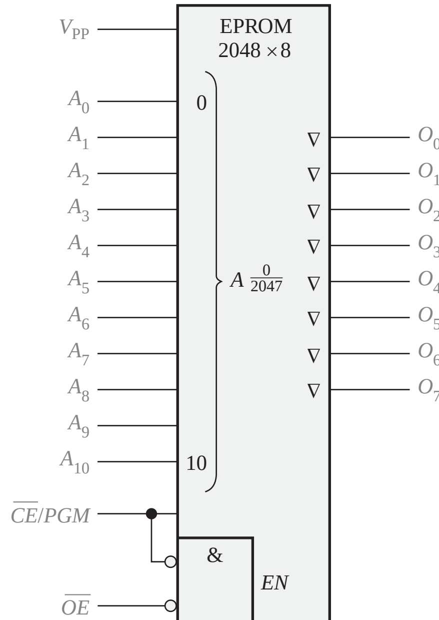


FIGURA 10.31 Símbolo lógico de una UV EPROM 2048 × 8.

Para leer la memoria, la entrada de habilitación de salida $(\overline{OE})$ tiene que estar a nivel BAJO y la entrada de habilitación de chip/programación $(\overline{CE}/PGM, chip\,enable/program)$ también a nivel BAJO. Para borrar los datos almacenados, el dispositivo se expone a una luz ultravioleta de alta intensidad que pasa a través de la tapa transparente. Una típica lámpara UV borrará los datos en unos 20 ó 25 minutos. Al igual que en la mayoría de las EPROM, tras el borrado todos los bits son 1. La luz ambiente normal contiene la longitud de onda ultravioleta adecuada para hacer que se realice el borrado tras un período de tiempo suficiente. Por tanto, la ventana transparente se ha de mantener tapada.

Para programar el dispositivo, se aplica una tensión continua alta a V_{PP}, estando $\overline{OE}$ a nivel ALTO. Los ocho bits de datos que van a ser programados en una dirección determinada se aplican a las salidas (Q_0 hasta Q_7) y se selecciona la dirección en las entradas A_0 hasta A_{10}. A continuación, se aplica un impulso a nivel ALTO a la entrada $\overline{CE}/PGM$. Las direcciones se pueden programar en cualquier orden.

En la Figura 10.32 se muestra un cronograma para el modo de programación. Normalmente, estas señales las produce un programador de dispositivos EPROM.

EEPROM. Las PROM borrables eléctricamente se pueden borrar y programar mediante impulsos eléctricos. Ya que se pueden grabar y borrar eléctricamente, las EEPROM se pueden programar y borrar rápidamente dentro del propio circuito final con fines de reprogramación.

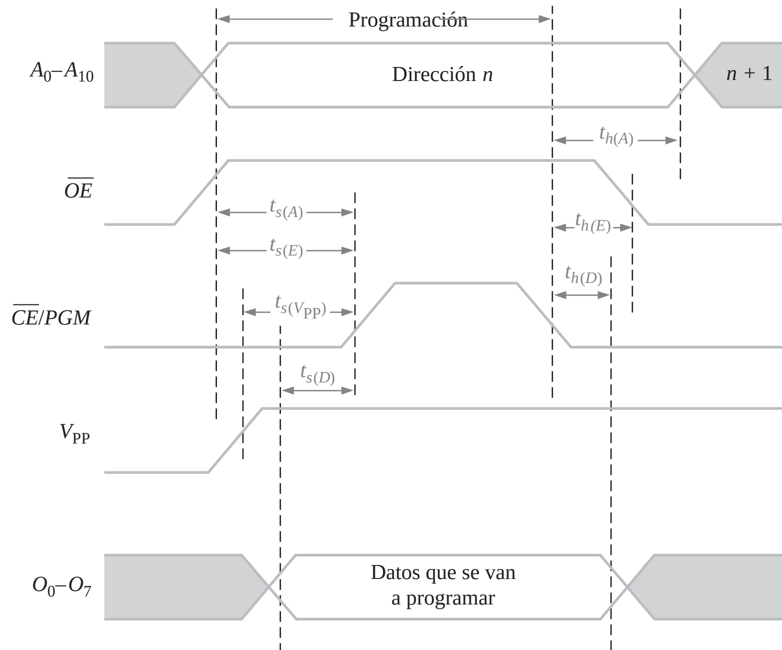


FIGURA 10.32 Diagrama de tiempos de un ciclo de programación de una UV EPROM 2048 × 8, que indica los tiempos de establecimiento (t_s) y de mantenimiento (t_h) críticos.

Los dos tipos de EEPROM son la MOS de puerta flotante y la de silicio óxido nitroso metal (MNOS, *Metal Nitride-Oxide Silicon*). La aplicación de una tensión en la puerta de control de la estructura de puerta flotante permite la eliminación y el almacenamiento de la carga en la puerta flotante.

REVISIÓN DE LA SECCIÓN 10.4

1. ¿En qué se diferencian las memorias PROM de las ROM?
2. Después del borrado, ¿todos los bits de una EPROM típica son 1 ó 0?
3. ¿Cuál es el modo normal de funcionamiento de una PROM?

10.5 MEMORIAS FLASH

La memoria ideal debería tener una alta capacidad de almacenamiento, ser no volátil, disponer de capacidad de lectura y escritura en el propio sistema, tener una velocidad de operación comparativamente rápida y ser efectiva en términos de coste. Las tecnologías de memorias tradicionales, como ROM, PROM, EPROM, EEPROM, SRAM y DRAM exhiben, cada una de ellas, una o más de estas características, pero ninguna tecnología las tiene todas, salvo la memoria flash.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Explicar las características básicas de una memoria flash. ■ Describir la operación básica de una célula de memoria flash. ■ Comparar las memorias flash con otros tipos de memoria

Las ***memorias flash*** son memorias de lectura/escritura de alta densidad (alta densidad equivale a gran capacidad de almacenamiento de bits) no volátiles, lo que significa que pueden almacenarse los datos indefinida-

mente en ausencia de alimentación. Estas memorias se utilizan frecuentemente en lugar de las unidades de disquete o de las unidades de disco duro de baja capacidad en las computadoras portátiles.

La característica de alta densidad significa que puede incluirse un gran número de celdas en un área de superficie dada del chip; es decir, cuanto más alta sea la densidad, más bits podrán almacenarse en un chip de un tamaño determinado. Esta alta densidad se consigue en las memorias flash con una célula de almacenamiento compuesta por un único transistor MOS de puerta flotante. El bit de datos se almacena como una carga o una ausencia de carga en la puerta flotante, dependiendo de si se desea almacenar un 0 o un 1.

Célula de memoria flash

La Figura 10.33 representa una célula monotransistor de una memoria flash. El transistor MOS de puerta apilada consta de una puerta de control y una puerta flotante, además del drenador y la fuente. La puerta flotante almacena electrones (carga) si se aplica la suficiente tensión a la puerta de control. *Se almacena un 0 cuando existe una cantidad significativa de carga y un 1 cuando la carga es menor, o inexistente*. La cantidad de carga presente en la puerta flotante determina si el transistor se activará y conducirá corriente del drenador a la fuente cuando se aplique una tensión de control durante una operación de lectura.

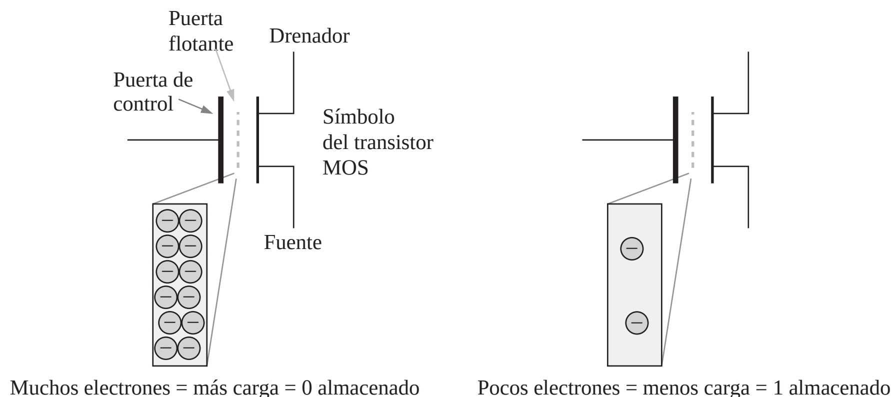


FIGURA 10.33 La célula de almacenamiento de una memoria flash.

Funcionamiento básico de la memoria flash

Hay tres operaciones principales en una memoria flash: la operación de *programación*, la operación de *lectura* y la operación de *borrado*.

Programación. Inicialmente, todas las células se encuentran en el estado 1, porque la carga fue eliminada de las células en una operación previa de borrado. La operación de programación añade electrones (carga) a la puerta flotante de aquellas células que deban almacenar un 0. No se añade carga a aquellas células que deban almacenar un 1. La aplicación a la puerta de control de una tensión suficientemente positiva con respecto a la fuente, durante la programación, atrae electrones a la puerta flotante, como indica la Figura 10.34. Una vez programada, una célula puede conservar la carga durante 100 años sin necesidad de aplicar una alimentación externa.

Lectura. Durante una operación de lectura, se aplica una tensión positiva a la puerta de control. La cantidad de carga presente en la puerta flotante de una célula determina si la tensión aplicada a la puerta de control activará, o no, el transistor. Si hay almacenado un 1, la tensión de la puerta de control es suficiente para activar el transistor. Si hay almacenado un 0, el transistor no se activará, porque la tensión de la puerta de control no

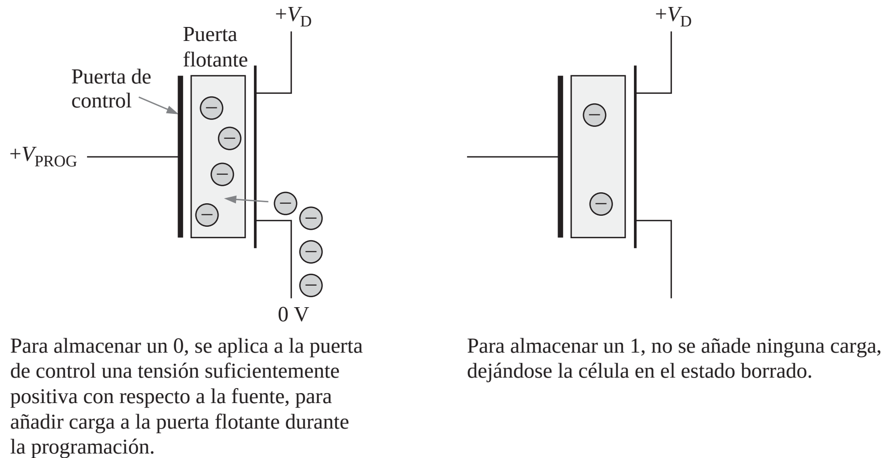


FIGURA 10.34 Ilustración simplificada del proceso de almacenamiento de un 0 o un 1 en una célula flash durante la operación de programación.

es suficiente para contrarrestar la carga negativa almacenada en la puerta flotante. Piense en la carga de la puerta flotante como en una fuente de tensión que se opone a la tensión aplicada a la puerta de control durante la lectura. Desde este punto de vista, la carga de la puerta flotante asociada a un 0 almacenado evita que la tensión de la puerta de control alcance el umbral de activación, mientras que la carga pequeña o nula asociada con un 1 almacenado permite a la tensión de la puerta de control exceder dicho umbral de activación.

Cuando el transistor se activa, existe corriente desde el drenador hacia la fuente del transistor de la célula. La presencia o ausencia de esta corriente es detectada para indicar un 1 o un 0, respectivamente. Esta idea básica se ilustra en la Figura 10.35.

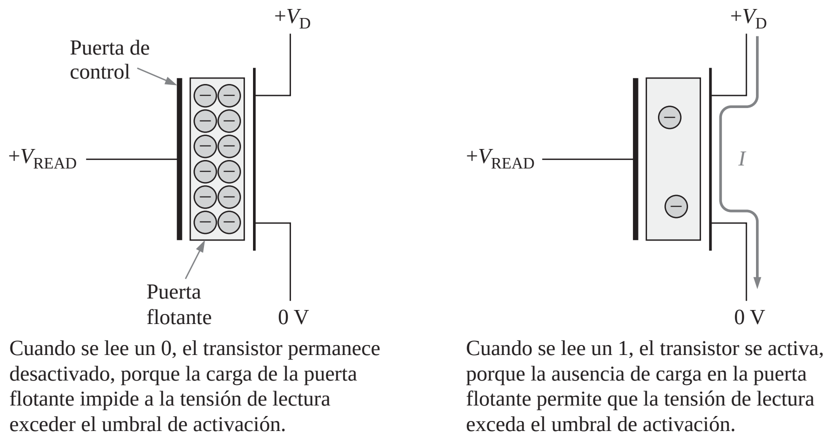
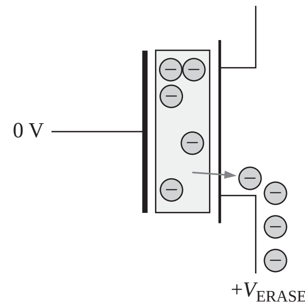


FIGURA 10.35 La operación de lectura de una célula flash de una matriz.

Borrado. Durante una operación de borrado, se elimina la carga de todas las células de memoria. Para ello, se aplica a la fuente del transistor una tensión suficientemente positiva con respecto a la puerta de control. Esta polaridad es precisamente la opuesta a la utilizada durante la programación. Esta tensión atrae a los electrones de la puerta flotante y hace que ésta se vacíe de carga, como se ilustra en la Figura 10.36. Las memorias flash siempre se borran antes de volver a ser programadas.



Para borrar una célula, se aplica a la fuente una tensión suficientemente positiva con respecto a la puerta de control, con el fin de extraer la carga de la puerta flotante durante la operación de borrado.

FIGURA 10.36 Ilustración simplificada del proceso de eliminación de la carga de una célula durante la operación de borrado.

Matriz básica de una memoria flash

La Figura 10.37 muestra una matriz simplificada de células de memoria flash. Sólo se accede a una fila cada vez. Cuando una celda de una línea de bit dada se activa (un 1 almacenado) durante una operación de lectura, existirá corriente a través de la línea de bit, lo que producirá una caída de tensión a través de la carga activa. Esta caída de tensión se compara con una tensión de referencia mediante un circuito comparador, generándose un nivel de salida que indica que hay un 1. Si hay un 0 almacenado, no hay corriente en la línea de bit o ésta es muy pequeña, generándose un nivel opuesto a la salida del comparador.

Comparación de las memorias flash con otras memorias

Vamos a comparar las memorias flash con los otros tipos de memorias con los que ya estamos familiarizados.

Flash frente a ROM, EPROM y EEPROM. Las memorias de sólo lectura son dispositivos de alta densidad y no volátiles. Sin embargo, una vez que se ha programado una ROM, su contenido no puede nunca alterarse. También la programación inicial es un proceso costoso y que consume tiempo.

Aunque la EPROM es una memoria de alta densidad no volátil, sólo se puede borrar extrayéndola del sistema y utilizando luz ultravioleta. Sólo se puede reprogramar utilizando un equipo especial.

La EEPROM tiene una estructura de celda más compleja que la ROM y la EPROM y su densidad no es tan alta, aunque puede reprogramarse sin sacarse del sistema. Debido a su densidad mucho menor, el coste por bit es mayor que en las ROM y EPROM.

Una memoria flash se puede reprogramar fácilmente dentro del sistema ya que, esencialmente, es un dispositivo de LECTURA/ESCRITURA. La densidad de una memoria flash es comparable a la de la ROM y la EPROM, ya que ambas utilizan celdas de un único transistor. Una memoria flash (al igual que una ROM, EPROM o EEPROM) es no volátil, lo que permite almacenar los datos indefinidamente sin alimentación.

Flash frente a SRAM. Como se ha explicado, las memorias estáticas de acceso aleatorio son dispositivos de LECTURA/ESCRITURA volátiles. Una SRAM requiere una alimentación constante para mantener los datos almacenados. En muchas aplicaciones, se utiliza una batería de reserva para evitar la pérdida de datos, si la fuente de alimentación principal se apaga. Sin embargo, puesto que siempre existe la posibilidad de que la batería falle, el mantenimiento indefinido de los datos en una SRAM no se puede garantizar. Como las celdas de memoria de una SRAM son, básicamente, un *latch* formado por varios transistores, la densidad es relativamente baja.

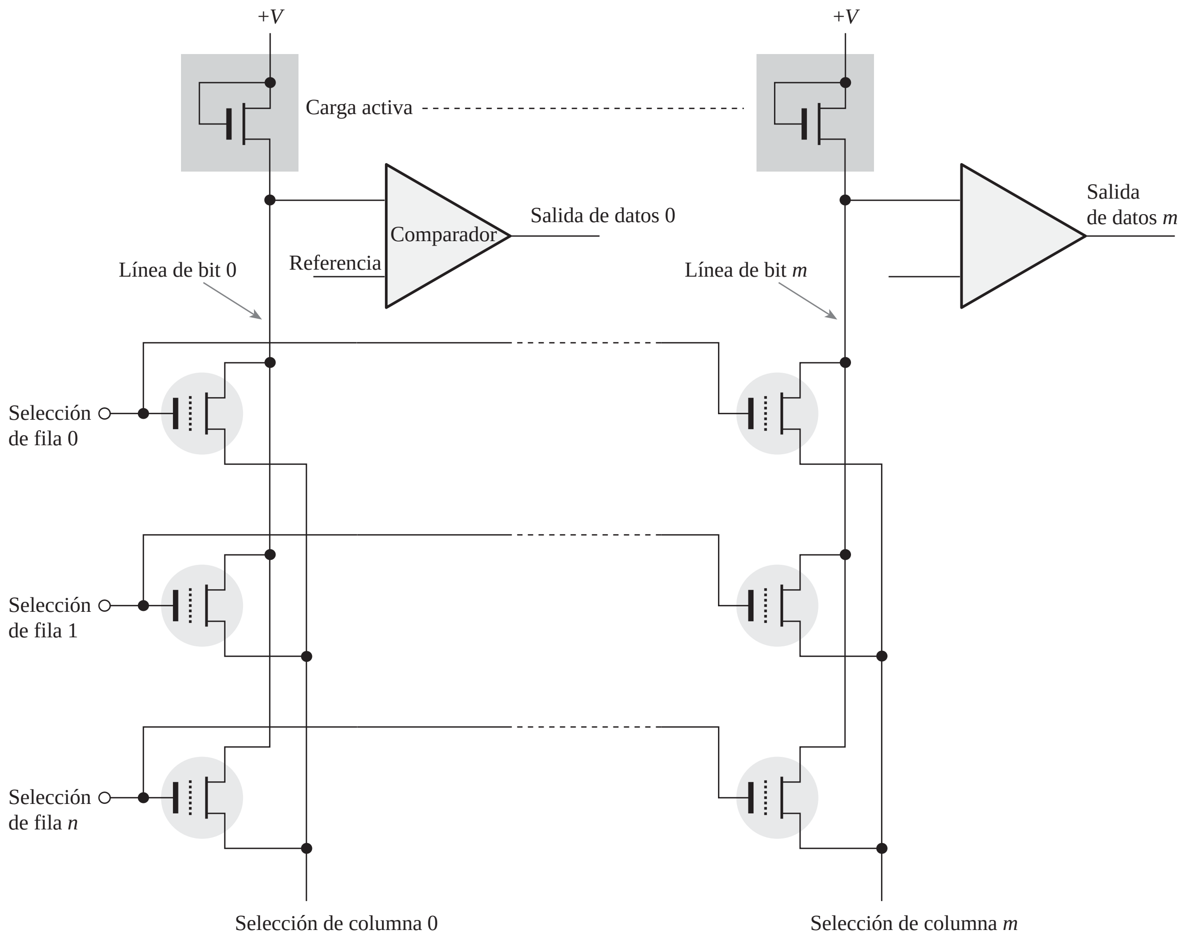


FIGURA 10.37 Matriz básica de una memoria flash.

Una memoria flash también es una memoria de LECTURA/ESCRITURA, pero, a diferencia de la SRAM, no es volátil. También, una memoria flash tiene una densidad mucho más alta que una SRAM.

Flash frente a DRAM. Las memorias de acceso aleatorio son dispositivos de LECTURA/ESCRITURA volátiles de alta densidad. Las DRAM no sólo requieren una alimentación constante para mantener los datos, sino que también los datos almacenados deben refrescarse frecuentemente. En muchas aplicaciones, debe utilizarse con la DRAM un almacenamiento de reserva, como por ejemplo un disco duro.

Las memorias flash tienen densidades más altas que las DRAM, porque una celda de memoria está formada por un transistor y no es necesario realizar el refresco, mientras que una celda DRAM es un transistor más un condensador que tiene que recargarse. Típicamente, una memoria flash consume menos potencia que una DRAM equivalente, y en muchas aplicaciones puede usarse para reemplazar un disco duro.

La Tabla 10.2 facilita un resumen de la comparación de las tecnologías de memorias.

REVISIÓN DE LA SECCIÓN 10.5

1. ¿Cuáles son los tipos de memoria no volátiles?
2. ¿Cuál es la ventaja principal de una memoria flash sobre una SRAM o una DRAM?
3. Enumerar los tres modos de operación de una memoria flash.

Tipo de memoria	No volátil	Alta densidad	Celda con un sólo transistor	Re-escribible en el sistema final
Flash	Sí	Sí	Sí	Sí
SRAM	No	No	No	Sí
DRAM	No	Sí	Sí	Sí
ROM	Sí	Sí	Sí	No
EPROM	Sí	Sí	Sí	No
EEPROM	Sí	No	No	Sí

TABLA 10.2 Comparación de los tipos de memoria.

10.6 EXPANSIÓN DE MEMORIAS

Las memorias disponibles se pueden ampliar para incrementar la longitud de palabra (número de bits en cada dirección) o la capacidad de palabra (número de direcciones diferentes), o ambas. La expansión de memoria se consigue añadiendo el número apropiado de chips de memoria a los buses de dirección, datos y control. También se presentan los módulos de expansión de memoria SIMM, DIMM y RIMM.

Al finalizar este capítulo, el lector deberá ser capaz de:

■ Definir la expansión de la longitud de palabra. ■ Mostrar cómo se amplia la longitud de palabra de una memoria. ■ Definir la expansión de la capacidad de palabra. ■ Mostrar cómo se amplia la capacidad de palabra de una memoria.

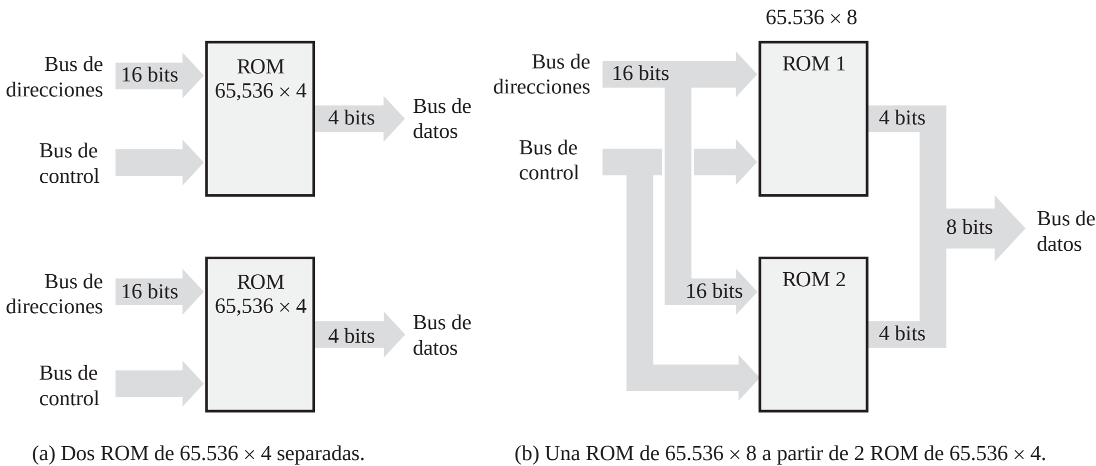


FIGURA 10.38 Expansión de dos ROM de 65.536 × 4 a una ROM de 65.536 × 8, que ilustra la expansión de la longitud de palabra.

Expansión de la longitud de palabra

Para aumentar la **longitud de palabra** de una memoria, el número de bits del bus de datos debe aumentarse. Por ejemplo, se puede conseguir una longitud de palabra de 8 bits utilizando dos memorias, teniendo cada una de ellas palabras de 4 bits, como se ilustra en la Figura 10.38(a). Como puede ver en la parte (b), el bus de direcciones de 16 bits se conecta normalmente a ambas memorias, de modo que la memoria combinada tenga el mismo número de direcciones ($2^{16} = 65.536$) que cada memoria individual. Los buses de datos de 4 bits de las dos memorias se combinan para formar un bus de datos de 8 bits. De este modo, cuando se selecciona una dirección, se producen ocho bits en el bus de datos, cuatro para cada memoria. El Ejemplo 10.2 muestra los detalles de la expansión de 65.536 × 4 a 65.536 × 8.

EJEMPLO 10.2

Expandir la ROM de 65.536 × 4 (64k × 4) de la Figura 10.39 para obtener una ROM de 64k × 8. Observe que "64k" es una abreviatura aceptada para designar 65.536. Pero, ¿por qué no se emplea "65k"? Posiblemente, porque 64 es una potencia de dos.

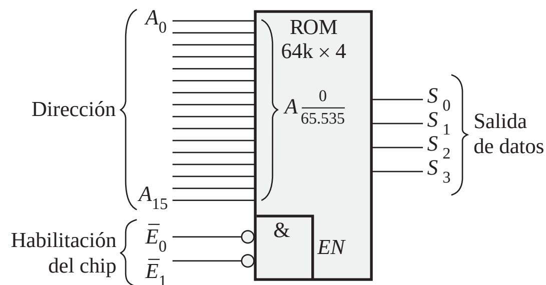


FIGURA 10.39 Una ROM de 64k × 4.

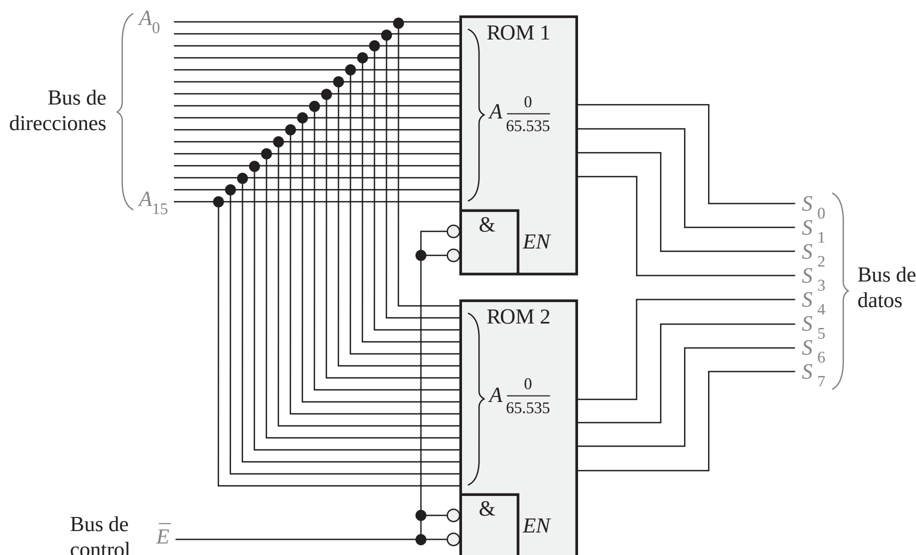


FIGURA 10.40

Solución Las dos ROM de 64k × 4 se conectan como se muestra en la Figura 10.40. Observe que se accede a una dirección específica en la ROM 1 y en la ROM 2 a la vez. Los cuatro bits de la dirección seleccionada de la ROM 1 y los cuatro bits correspondientes de la ROM 2 se llevan en paralelo para formar una palabra de 8 bits en el bus de datos. Observe también que un nivel BAJO en la línea de habilitación de chip $\overline{E}$, que actúa como un bus de control simple, activa *ambas* memorias.

Problema relacionado Describir la manera de expandir una ROM de 64k × 1 para obtener una ROM de 64k × 8.

EJEMPLO 10.3

Utilizar las memorias del Ejemplo 10.2 para formar una ROM de 64k × 16.

Solución En este caso se necesita una memoria que almacene 65.356 palabras de 16 bits. Se requieren cuatro ROM de 64k × 4 para realizar el trabajo, como se muestra en la Figura 10.41.

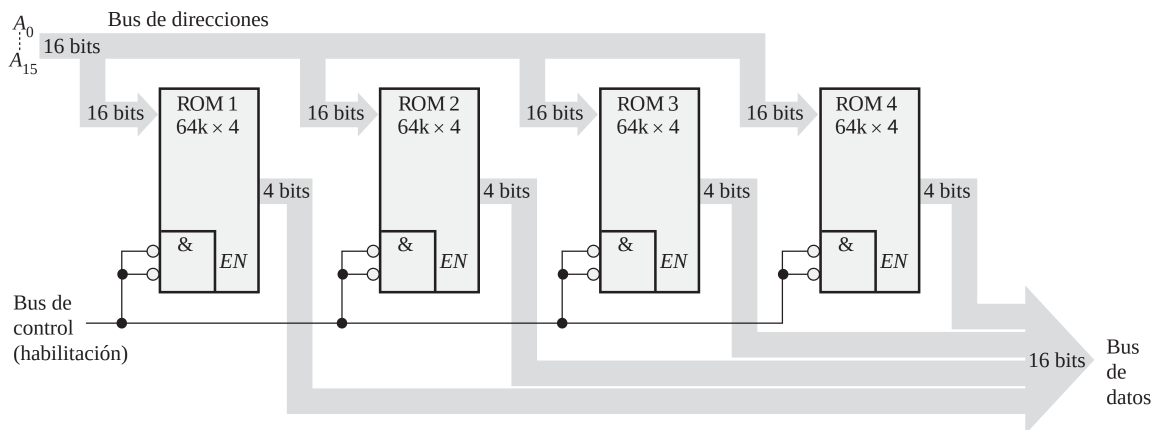


FIGURA 10.41

Problema relacionado ¿Cuántas ROM de 64k × 1 se requerirían para implementar la memoria mostrada en la Figura 10.41?

Una ROM sólo tiene salidas de datos, pero una RAM tiene entradas y salidas de datos. Para realizar la expansión de la longitud de palabra en una RAM (SRAM o DRAM), las entradas *y* salidas de datos forman el bus de datos. Puesto que las líneas de entrada de datos y las correspondientes líneas de salida de datos deben conectarse juntas, se requieren buffers triestado. La mayoría de las RAM proporcionan circuitería de tres estados interna. La Figura 10.42 ilustra la expansión de la RAM para incrementar la longitud de palabra.

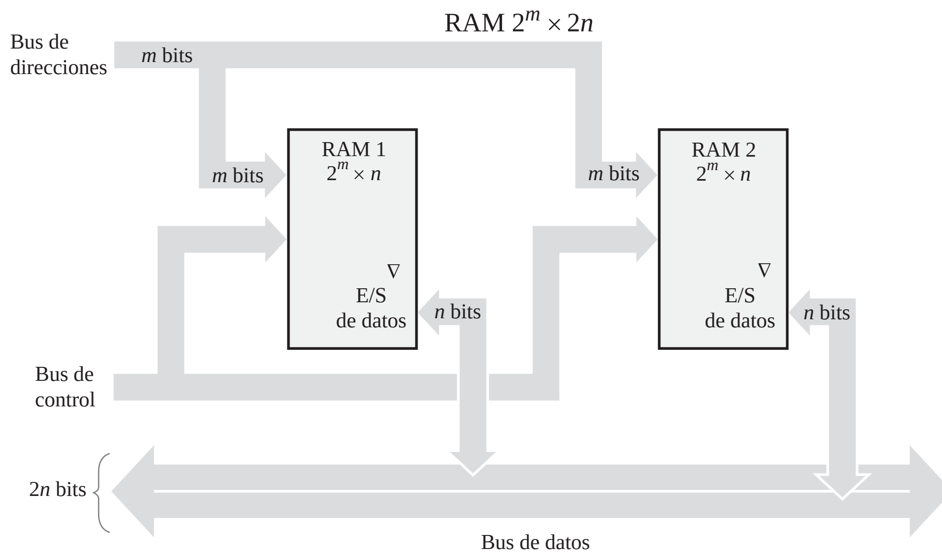


FIGURA 10.42 Ilustración de la expansión de palabra, con dos RAM de $2^m \times n$ unidas para formar una RAM de $2^m \times 2n$.

EJEMPLO 10.4

Utilizar memorias SRAM de 1M × 4 para crear una SRAM de 1M × 8.

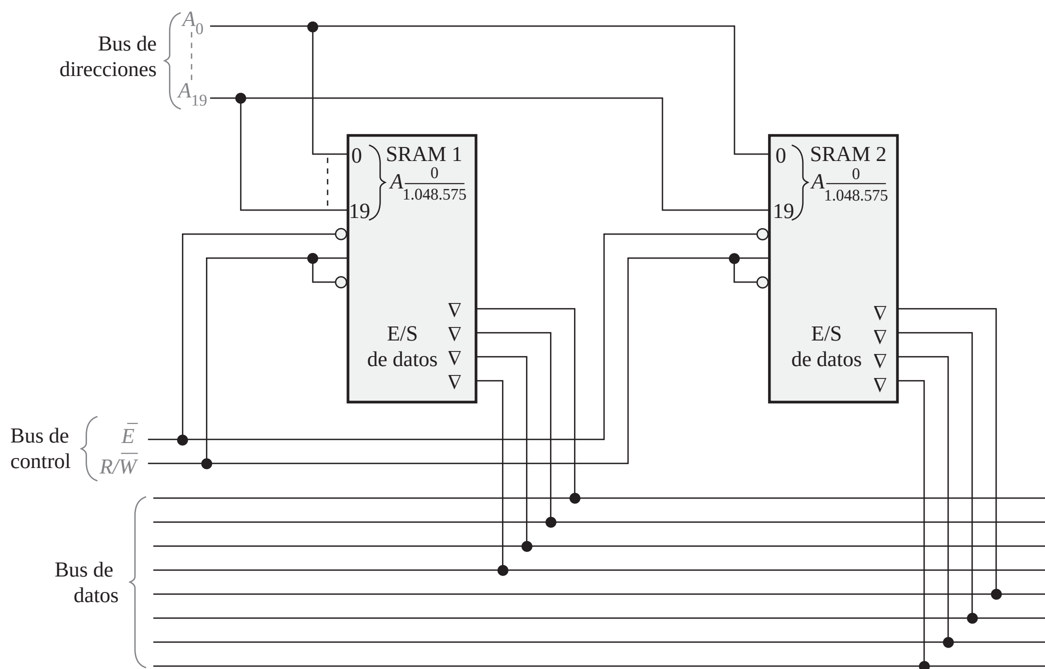
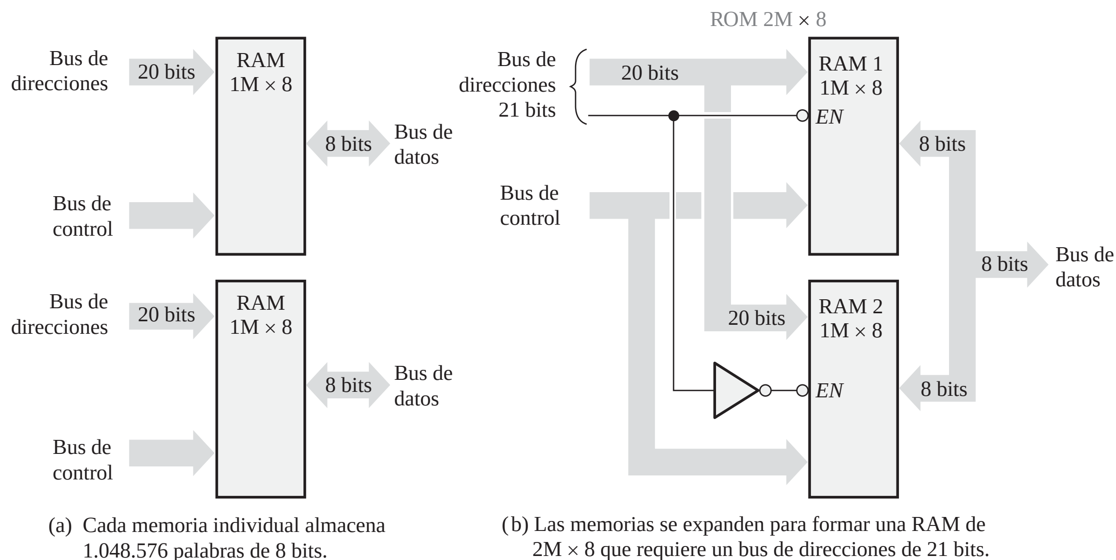


FIGURA 10.43

Solución	Se conectan dos SRAM de 1M × 4 como muestra el diagrama de bloques simplificado de la Figura 10.43.
Problema relacionado	Utilizar memorias SRAM de 1M × 8 para crear una SRAM de 1M × 16.

Expansión de la capacidad de palabra

Cuando las memorias se amplían para incrementar la **capacidad de palabra**, el *número de direcciones se aumenta*. Para conseguir este incremento, el número de bits de dirección se debe aumentar como se ilustra en la Figura 10.44, en la que se expanden dos RAM de 1M × 8 para formar una memoria de 2M × 8.



(a) Cada memoria individual almacena 1.048.576 palabras de 8 bits.

(b) Las memorias se expanden para formar una RAM de 2M × 8 que requiere un bus de direcciones de 21 bits.

FIGURA 10.44 Expansión de la capacidad de palabra.

Cada memoria individual tiene 20 bits de dirección para seleccionar 1.048.576 direcciones, como se muestra en la parte (a). La memoria expandida tiene 2.097.152 direcciones y, por tanto, requiere 21 bits de dirección, como se muestra en la parte (b). El bit de dirección número 21 se utiliza para activar el chip de memoria adecuado. El bus de datos de la memoria expandida resultante sigue siendo de ocho bits. En el Ejemplo 10.5 se ilustran los detalles de esta expansión.

EJEMPLO 10.5

Utilizar memorias RAM de 512k × 4 para implementar una memoria de 1M × 4.

Solución La expansión del direccionamiento se consigue conectando la entrada de habilitación de chip ($\overline{E}_0$) al vigésimo bit de dirección (A_{19}) como muestra la Figura 10.45. La entrada $\overline{E}_1$ se usa como entrada de habilitación común a las dos memorias. Cuando el vigésimo bit de dirección (A_{19}) está a nivel BAJO, se selecciona la RAM 1 (la RAM 2 está desactivada), y los 19 bits de dirección de menor orden (A_0-A_{18}) permiten acceder a las direcciones de la RAM 1.

Cuando el vigésimo bit de dirección (A_{19}) está a nivel ALTO, la RAM 2 se activa debido al nivel BAJO en la salida del inversor (la RAM 1 está desactivada), y los 19 bits de dirección de menor orden (A_0-A_{18}) permiten acceder a cada una de las direcciones de la RAM 2.

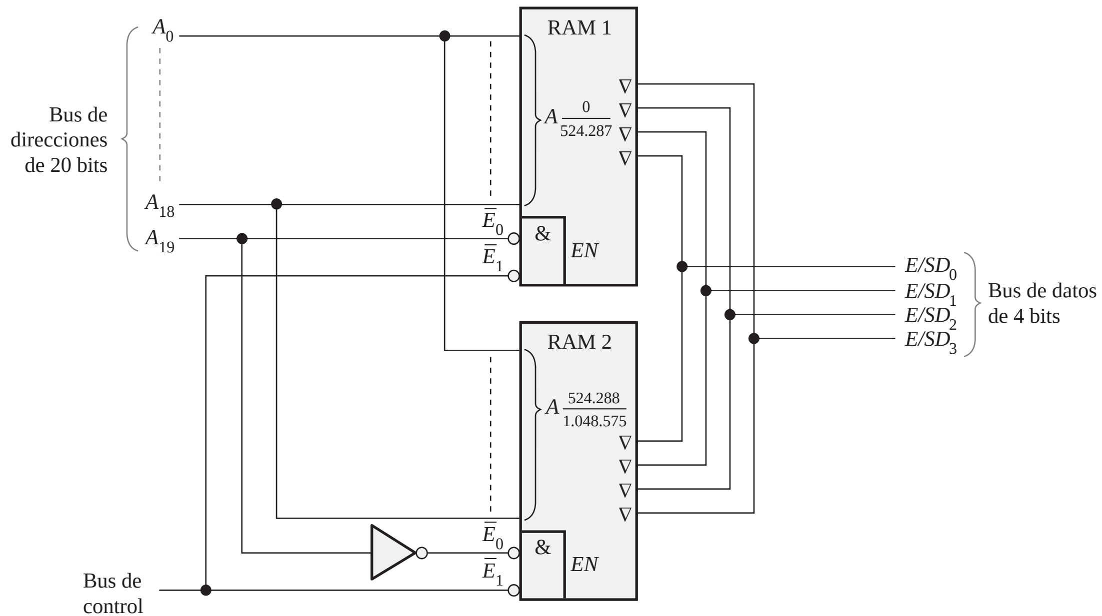


FIGURA 10.45

Problema relacionado ¿Cuáles son los rangos de direcciones de la RAM 1 y de la RAM 2 de la Figura 10.45?

Módulos de memoria

Las memorias se suelen suministrar en forma de módulos de memoria de terminal simple (*single in-line memory module*, **SIMM**) o como módulos de memoria de terminal doble (*dual in-line memory module*, **DIMM**). Los módulos SIMM y los más recientes módulos DIMM son pequeñas tarjetas de circuito impreso en las que se montan chips de memoria con las entradas y salidas conectadas a un conector de borde situado en la parte inferior de la tarjeta de circuito. Los módulos DIMM son generalmente más rápidos, pero sólo pueden ser instalados en las máquinas más modernas, que hayan sido diseñadas para admitirlos.

Los módulos SIMM se clasifican en módulos de 30 y de 72 pines. Ambos tipos se ilustran en la Figura 10.46. Aunque las capacidades de memoria disponibles para los módulos SIMM pueden variar entre 256 KB y 32 MB, la diferencia fundamental entre las dos configuraciones de pines es el tamaño del bus de datos. Generalmente, los módulos SIMM de 30 pines están diseñados para buses de datos de 8 bits, necesitándose más módulos SIMM para poder manejar más bits de datos. Los módulos SIMM de 72 pines admiten un bus de datos de 32 bits, por lo que hacen falta un par de módulos SIMM para los buses de datos de 64 bits.

Los módulos DIMM tienen un aspecto similar a los módulos SIMM, pero proporcionan una mayor densidad de memoria con sólo un incremento relativamente pequeño en el tamaño físico. La diferencia clave

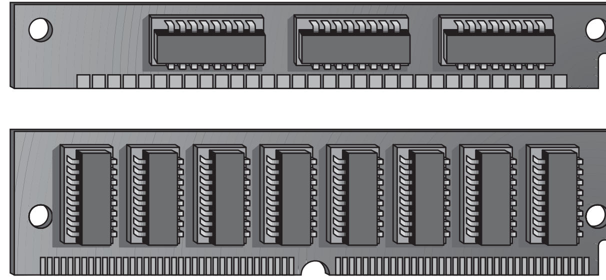


FIGURA 10.46 Módulos SIMM de 30 y de 72 pines.

estriba en que los módulos DIMM distribuyen los pines de entrada y salida en ambos lados de la tarjeta de circuito impreso, mientras que los módulos SIMM sólo emplean uno de los lados. Las configuraciones normales de los módulos DIMM son las de 72 pines, 100 pines, 144 pines y 168 pines, con las que se pueden emplear buses de datos de 32 y de 64 bits. Por regla general, la capacidad de los módulos DIMM va de 4 a 512 MB.

Los módulos SIMM y DIMM se enchufan en zócalos situados en una tarjeta de sistema, como los ilustrados en la Figura 10.47, siendo lo normal que haya varios zócalos disponibles para la expansión de memoria. Por supuesto, los zócalos para los módulos SIMM y DIMM son diferentes y no son intercambiables.

Otro módulo estándar de memoria, similar al módulo DIMM, pero con un bus de mayor velocidad, es el módulo RIMM (*Rambus In Line Memory Module*). Muchas computadoras portátiles utilizan una variante del módulo DIMM llamada SODIMM, que es de menor tamaño, tiene 144 pines y una capacidad de hasta 256 MB.

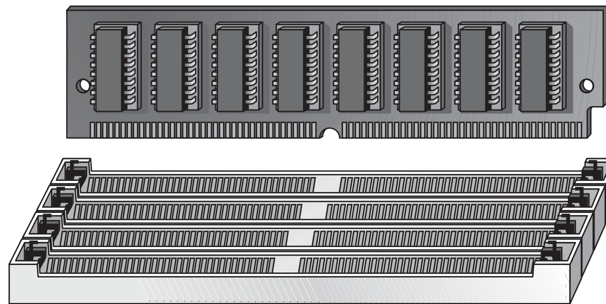


FIGURA 10.47 Un módulo SIMM/DIMM insertado en un zócalo de una tarjeta de un sistema.

CONSEJOS PRÁCTICOS

Los componentes de memoria son extremadamente sensibles a la electricidad estática. Tome las siguientes precauciones cuando esté manejando chips de memoria o módulos de tipo SIMM y DIMM:

- Antes de manejar los componentes, descargue la electricidad estática de su cuerpo tocando una superficie puesta a masa, o lleve una muñequera de puesta a tierra con una resistencia de alto valor, si dispone de una. Una toma de tierra cómoda y fiable es la que puede encontrarse en los enchufes eléctricos.
- No extraiga los componentes de sus bolsas antiestáticas hasta que esté listo para instalarlos.
- No deposite componentes sobre las bolsas antiestáticas, porque sólo el interior de las mismas es antiestático.

CONSEJOS PRÁCTICOS

- Cuando esté manejando módulos SIMM o DIMM, sosténgalos por los bordes o por la abrazadera metálica de montaje. No toque los componentes de las placas ni los terminales del conector de borde.
- No deslice nunca un componente sobre ningún tipo de superficie.
- Evite la presencia de plástico, vinilo, espuma de poliestireno y nilón en el área de trabajo.

Cuando instale módulos SIMM o DIMM, siga estos pasos:

1. Alinee las muescas de las tarjetas del módulo SIMM o DIMM con las muescas del zócalo de memoria.
2. Presione firmemente el módulo hasta haberlo encajado en el zócalo.
3. Generalmente, los cierres en ambos lados del zócalo se ajustarán al insertar completamente el módulo. Estos cierres también permiten liberar el módulo, para poderlo sacar del zócalo.

REVISIÓN DE LA SECCIÓN 10.6

1. ¿Cuántas RAM de 16k × 1 se requieren para conseguir una memoria con una capacidad de palabra de 16k y una longitud de palabra de ocho bits?
2. Para expandir la memoria de 16k × 8 de la cuestión anterior en una organización de 32k × 8, ¿cuántas RAM de 16k × 1 se requieren?
3. ¿Qué significa SIMM?
4. ¿Qué significa DIMM?
5. ¿Qué significa el término RIMM?

10.7 TIPOS ESPECIALES DE MEMORIAS

En esta sección, se cubren la memoria FIFO (*First In-First Out*, primero en entrar primero en salir), la memoria LIFO (*Last In-First Out*, último en entrar primero en salir), la pila de memoria y el dispositivo de memoria de acoplamiento carga.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Describir una memoria FIFO. ■ Describir una memoria LIFO. ■ Utilizar las pilas de memoria. ■ Explicar cómo se usa una parte de una RAM como pila de memoria. ■ Describir una memoria básica CCD.

Memorias FIFO (*First In-First Out*)

Este tipo de memoria está formado por una disposición de registros de desplazamiento. El término ***FIFO*** hace referencia al funcionamiento básico de este tipo de memoria, en la que el primer bit de datos que se escribe es el primero que se lee.

En la Figura 10.48 se ilustra una diferencia importante entre un registro de desplazamiento convencional y un registro FIFO. En un registro convencional, un bit de datos se desplaza a través del registro sólo cuando se introducen nuevos datos; en un registro FIFO, un bit de datos atraviesa el registro hasta situarse en la posición de bit más a la derecha que esté vacía.

La Figura 10.49 es el diagrama de bloques de una memoria serie FIFO. Esta memoria en particular tiene cuatro registros de datos serie de 64 bits y un registro de control de 64 bits (registro de marca). Cuando los datos se introducen mediante un impulso de desplazamiento de entrada, automáticamente, bajo el control del

Registro de desplazamiento convencional					
Entrada	X	X	X	X	Salida
0	0	*X*	*X*	*X*	⟶
1	1	0	*X*	*X*	⟶
1	1	1	0	*X*	⟶
0	0	1	1	0	⟶

X = bits de datos desconocidos.
En un registro de desplazamiento convencional, los datos permanecen a la izquierda hasta que son desplazados por medio de datos adicionales.

Registro de desplazamiento FIFO					
Entrada	—	—	—	—	Salida
0	—	—	—	0	⟶
1	—	—	1	0	⟶
1	—	1	1	0	⟶
0	0	1	1	0	⟶

— = posiciones vacías.
En un registro de desplazamiento FIFO, los datos "van cayendo" hacia la derecha.

FIGURA 10.48 Comparación del funcionamiento de un registro convencional y uno FIFO.

registro de marca, se mueven a la posición vacía más próxima a la salida. Los datos no pueden avanzar a las posiciones que están ocupadas. Sin embargo, cuando un bit de datos se desplaza mediante un impulso de desplazamiento de salida, los bits de datos que están en los registros automáticamente se mueven a la posición siguiente hacia la salida. En una memoria FIFO asíncrona, los datos se desplazan hacia fuera independientemente de la entrada de datos, utilizando dos relojes separados.

Aplicaciones de una FIFO

Un área de aplicación importante del registro FIFO es el caso en que dos sistemas con velocidades diferentes tienen que comunicarse. Los datos pueden entrar en un registro FIFO a una velocidad y salir a otra velocidad distinta. La Figura 10.50 muestra cómo debe emplearse un registro FIFO en estas situaciones.

Memorias LIFO (*Last In-First Out*)

Las memorias **LIFO** se encuentran en aplicaciones que utilizan microprocesadores y otros sistemas de computación. Permiten almacenar datos y luego extraerlos en orden inverso; es decir, el último byte de datos almacenado es el primer byte de datos que se recupera.

Pilas de registros. Comúnmente, una memoria LIFO se denomina pila *push-down*. En algunos sistemas, se implementa con un grupo de registros, como muestra la Figura 10.51. Una pila puede estar formada por cualquier número de registros, pero el registro superior se denomina *tope de la pila.*

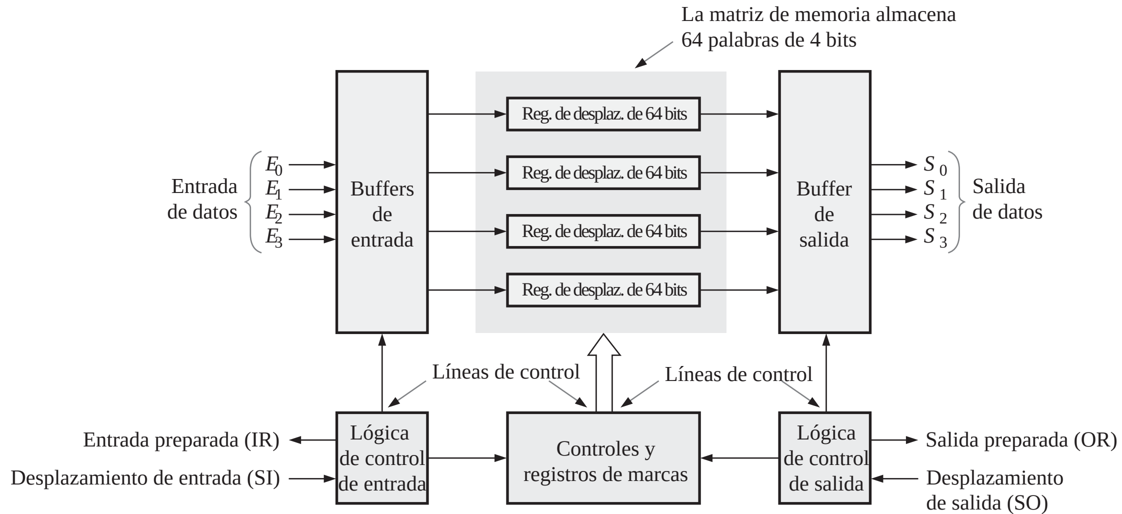


FIGURA 10.49 Diagrama de bloques de una memoria FIFO serie típica.

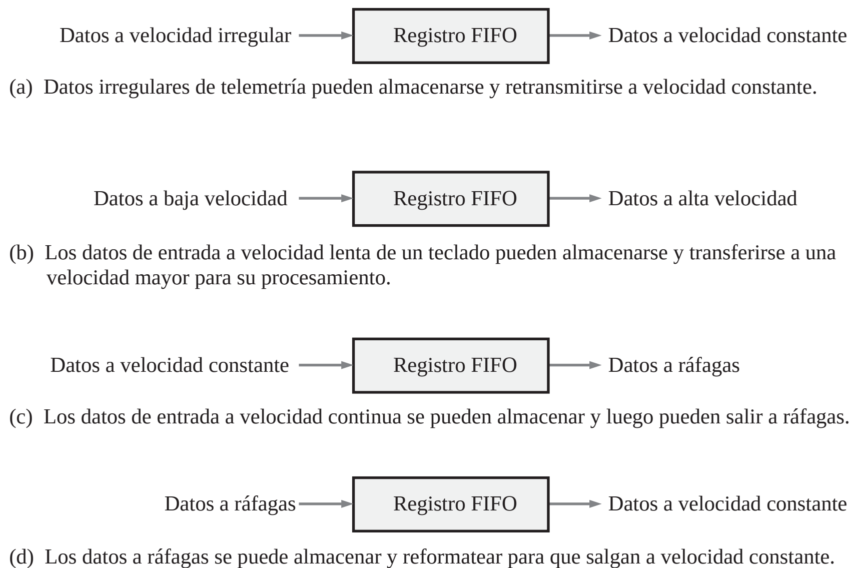


FIGURA 10.50 Ejemplos de registro FIFO para aplicaciones de control de la velocidad de transmisión de datos.

Para ilustrar el principio de estas memorias, se carga en paralelo un byte de datos en el tope de la pila. Cada sucesivo byte empuja al anterior al registro siguiente. Este proceso se muestra en la Figura 10.52. Observe que cada nuevo dato siempre se carga en el registro superior, y los bytes previamente almacenados son empujados hacia abajo dentro de la pila. El nombre de *pila push-down* (pila de empuje hacia abajo) viene de esta característica.

Los bytes de datos se recuperan en orden inverso. El último byte introducido siempre está en el registro superior de la pila, por lo que, cuando sale de la pila, los demás bytes saltan a las siguientes posiciones superiores. Este proceso se ilustra en la Figura 10.53.

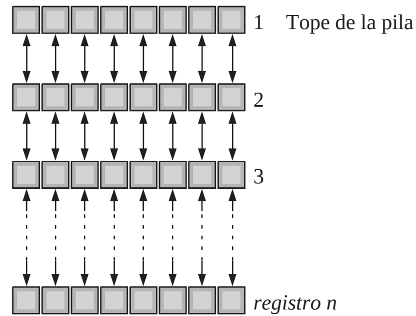


FIGURA 10.51 Pila de registros.

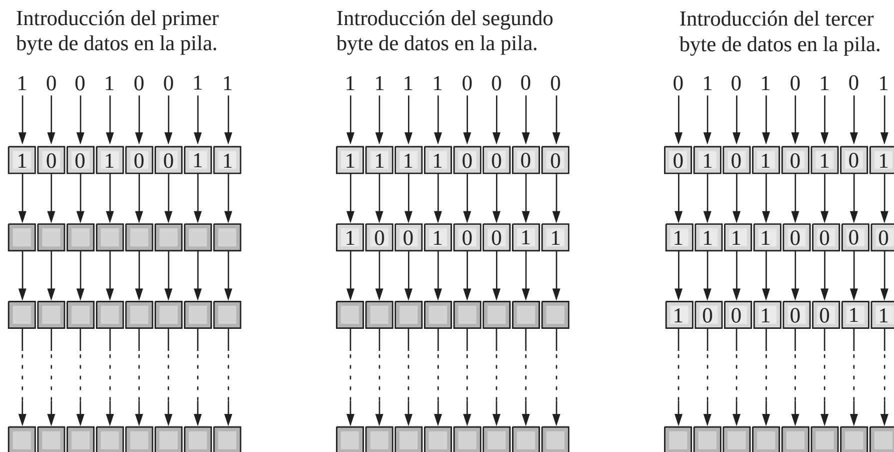


FIGURA 10.52 Ilustración simplificada del proceso de introducción de datos en la pila.

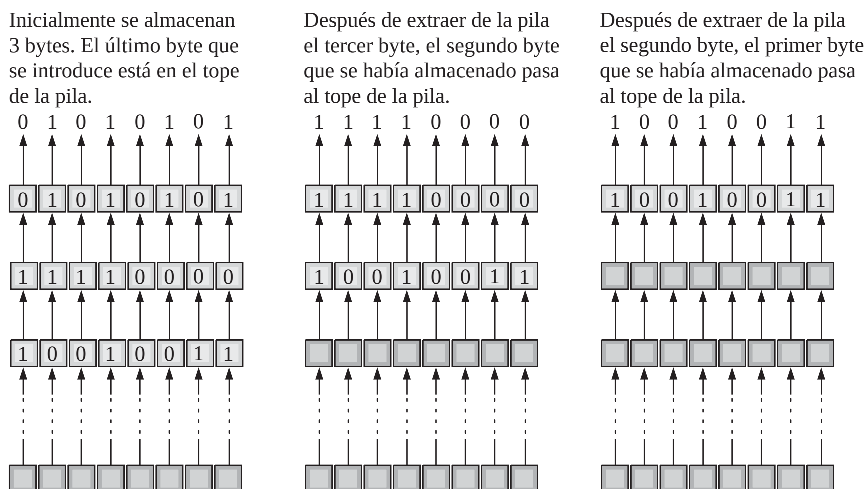


FIGURA 10.53 Ilustración simplificada de extracción de datos de la pila

Pila RAM. Otra forma de implementar una memoria LIFO, en algunos sistemas basados en microprocesador, es disponer una sección de la RAM como pila, mejor que utilizar un conjunto de registros dedicados. Como hemos visto, en un registro de pila, los datos se desplazan de arriba a abajo desde una posición a la siguiente. En una pila RAM, los datos en sí no se desplazan, sino que es el tope de la pila lo que se mueve bajo el control de un registro denominado puntero de pila.

Consideremos una memoria de acceso aleatorio organizada en bytes, es decir, en la que cada dirección contiene ocho bits, como muestra la Figura 10.54. La dirección binaria 0000000000001111, por ejemplo, puede escribirse en hexadecimal como 000F. Una dirección de 16 bits puede tener un valor hexadecimal *mínimo* de 0000_{16} y un valor *máximo* de $FFFF_{16}$. Utilizando esta notación, una matriz de memoria de 64 kB se puede representar como se muestra en la Figura 10.54. La dirección de memoria más baja es 0000_{16} y la dirección más alta es $FFFF_{16}$.

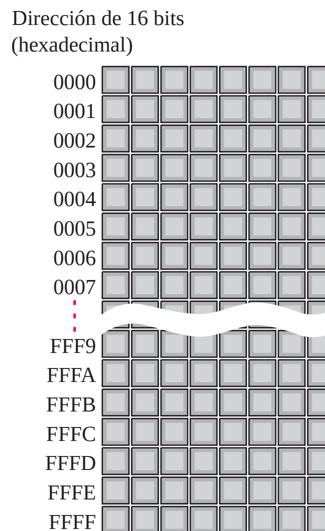
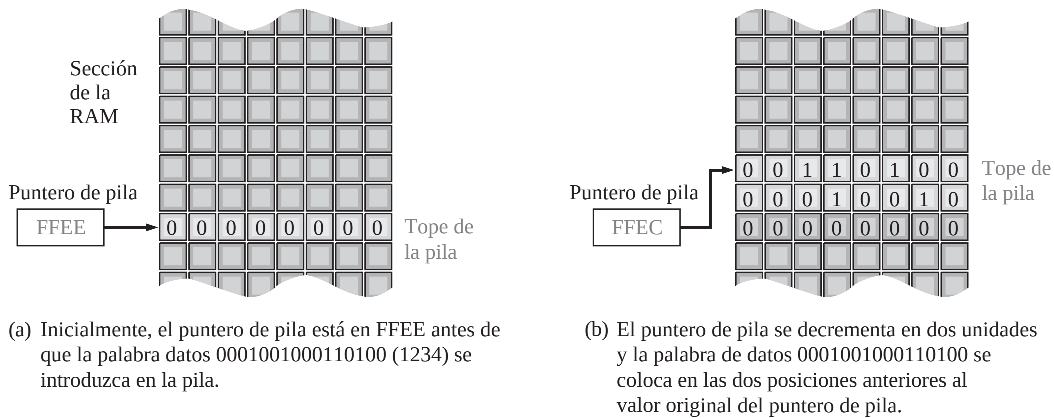


FIGURA 10.54 Representación de una memoria de 64 kB con direcciones de 16 bits expresadas en hexadecimal.

Ahora, supongamos que se reserva una sección de la RAM para utilizarla como pila. Un registro independiente especial, denominado puntero de pila, contiene la dirección del registro superior de la pila, tal y como se muestra en la Figura 10.55. Se utiliza una representación hexadecimal de 4 dígitos para las direcciones binarias. En la figura, las direcciones se han elegido arbitrariamente, con el único fin de servir de ilustración.

Ahora vamos a ver cómo se introducen los datos en la pila. Inicialmente, el puntero de pila se encuentra en la dirección $FFEE_{16}$, que es el tope de la pila, como se indica en la Figura 10.55(a). El puntero de pila se decrementa en dos unidades y toma el valor $FFEC_{16}$. Esto hace que el tope de la pila se mueva a una dirección de memoria inferior, como se ve en la Figura 10.55(b). Observe que el tope de la pila no es estacionario como en el caso de la pila de registros fijos, sino que se mueve hacia abajo (direcciones más bajas) por la RAM, según se almacenan las palabras de datos. La Figura 10.55(b) muestra dos bytes (una palabra de datos) que se introducen en la pila. Después de que se almacena la palabra, el tope de la pila está en $FFEC_{16}$.

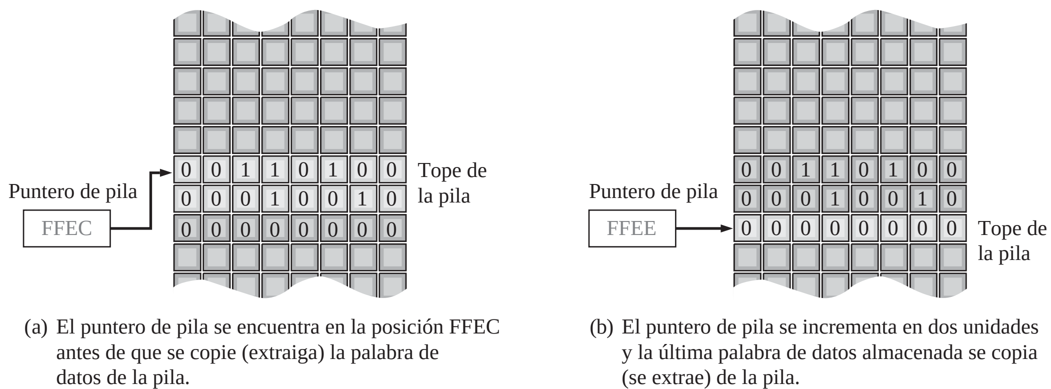
La Figura 10.56 ilustra la operación de extracción de los datos de la pila RAM. La última palabra de datos almacenada en la pila se lee en primer lugar. El puntero de pila, que está en la posición FFEC, se incrementa en dos unidades para apuntar a la dirección $FFEE_{16}$ y se realiza una operación de extracción, como puede verse en la parte (b) de la figura. Tenga en mente que las memorias RAM, cuando se leen, no son destructivas, por lo que los datos permanecerán almacenados en la memoria después de la operación de extracción. Una palabra de datos sólo se destruye cuando se escribe sobre ella.



(a) Inicialmente, el puntero de pila está en FFEE antes de que la palabra datos 0001001000110100 (1234) se introduzca en la pila.

(b) El puntero de pila se decrementa en dos unidades y la palabra de datos 0001001000110100 se coloca en las dos posiciones anteriores al valor original del puntero de pila.

FIGURA 10.55 Ilustración de la operación de introducción de datos en una pila RAM.



(a) El puntero de pila se encuentra en la posición FFEC antes de que se copie (extraiga) la palabra de datos de la pila.

(b) El puntero de pila se incrementa en dos unidades y la última palabra de datos almacenada se copia (se extrae) de la pila.

FIGURA 10.56 Ilustración de la operación de extracción de datos de la pila RAM.

Memorias CCD

La memoria **CCD** (*charge-coupled device*, dispositivo de acoplamiento de carga) almacena los datos como cargas de condensador. Sin embargo, a diferencia de la RAM dinámica, la celda de almacenamiento no incluye un transistor. La principal ventaja de estas memorias CCD es su alta densidad.

La memoria CCD está formada por largas filas de condensadores semiconductores, denominados *canales*. Los datos se introducen en serie en el canal, depositando una pequeña carga en el condensador si se trata de un 0, y una carga grande si es un 1. Después, estas cargas se desplazan a lo largo del canal mientras que se introducen más datos, de acuerdo con las señales de reloj.

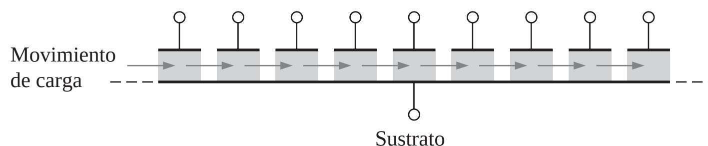


FIGURA 10.57 Un canal CCD (charge-coupled device).

Como en el caso de la DRAM, las cargas se deben refrescar periódicamente. Este proceso se realiza desplazando las cargas en serie a través de un circuito de refresco. La Figura 10.57 muestra el concepto básico de un canal CCD. Como los datos se desplazan en serie a través de los canales, estas memorias tienen un tiempo de acceso relativamente grande. Las matrices CCD se usan en algunas cámaras modernas, para captura de imágenes de vídeo en forma de carga de luz inducida.

REVISIÓN DE LA SECCIÓN 10.7

1. ¿Qué es una memoria FIFO?
2. ¿Qué es una memoria LIFO?
3. Explicar la operación de introducción de datos en una pila de memoria.
4. Explicar la operación de extracción de datos de una pila de memoria.
5. ¿De qué es abreviatura el término CCD?

10.8 MEMORIAS ÓPTICAS Y MAGNÉTICAS

En esta sección nos ocuparemos de los fundamentos de los discos magnéticos, cintas magnéticas, discos magneto-ópticos y discos ópticos. Estos medios de almacenamiento son muy importantes, especialmente en aplicaciones de computadoras, en las que se utiliza almacenamiento masivo no volátil de datos y programas.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Describir un disco duro magnético. ■ Describir los discos flexibles. ■ Describir los discos duros extraibles. ■ Explicar el principio básico de los discos magneto-ópticos. ■ Definir los discos CD-ROM, CD-R y CD-RW. ■ Describir una memoria WORM. ■ Describir el DVD-ROM.

Almacenamiento magnético

Disco duro magnético. Las computadoras emplean el disco duro como dispositivo interno de almacenamiento masivo. Los ***discos duros*** son “placas” rígidas de aleación de aluminio o de una mezcla de vidrio y cerámica recubiertos con una capa magnética. Hay disponibles unidades de disco duro, con dos tamaños principales de diámetro, 5,25 y 3,5 pulgadas, aunque también existen de 2,5 y 1,75 pulgadas. Las unidades de disco duro se sellan herméticamente para mantener al disco libre de polvo.

Normalmente, se apilan dos o más discos sobre un eje o pivote común, que hace que el conjunto gire a una velocidad de miles de revoluciones por minuto (rpm). Existe una separación entre cada disco, con el fin de permitir el montaje de un cabezal de lectura-escritura en el extremo del brazo accionador, como se muestra en la Figura 10.58. Hay un cabezal de lectura-escritura en cada cara del disco, ya que los datos se graban en ambas caras de la superficie del disco. El brazo accionador de la unidad sincroniza todos los cabezales de lectura-escritura para mantenerlos perfectamente alineados cuando se desplazan por la superficie del disco, con una separación de sólo una fracción de milímetro con respecto al disco. Una pequeña partícula de polvo podría hacer que un cabezal se rompiera, dañando como consecuencia la superficie del disco.

Principios básicos del cabezal de lectura-escritura. El disco duro es un dispositivo de acceso aleatorio, ya que puede recuperar datos almacenados en cualquier lugar del disco, en cualquier orden. En la Figura 10.59 se presenta un diagrama simplificado de la operación de lectura-escritura en la superficie magnética. La dirección o polarización de las partículas magnéticas sobre la superficie del disco se controla mediante la dirección de las líneas de flujo magnético (campo magnético) producidas por el cabezal de escritura, según la dirección de un impulso de corriente en el devanado. Este flujo magnético magnetiza un pequeño punto de la superficie del disco en la dirección del campo magnético. Un punto magnetizado con una cierta polaridad representa un

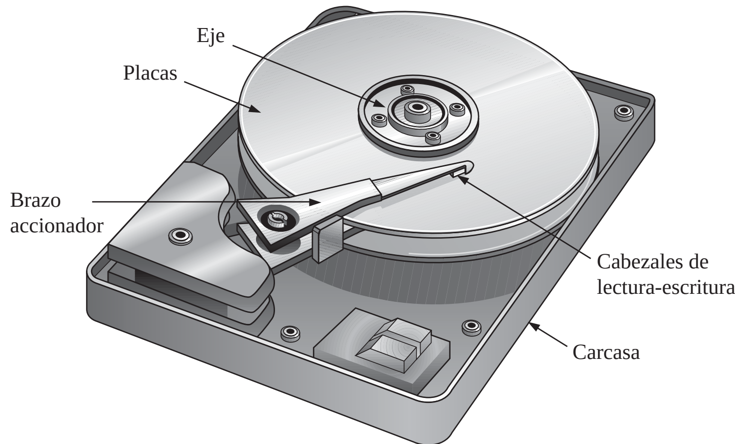


FIGURA 10.58 Esquema simplificado de una unidad de disco duro.

1 binario, y uno de polaridad opuesta representa un 0 binario. Una vez que se ha magnetizado un punto de la superficie, permanece en dicho estado hasta que se escribe sobre él un campo magnético opuesto.

Cuando un cabezal de lectura pasa por una superficie magnetizada, los puntos magnetizados generan campos magnéticos en el cabezal de lectura, lo que provoca impulsos de tensión en el devanado. La polaridad de estos impulsos depende de la dirección del punto magnetizado e indica si el bit almacenado es un 1 o un 0. A menudo, los cabezales de lectura y de escritura se combinan en una única unidad.

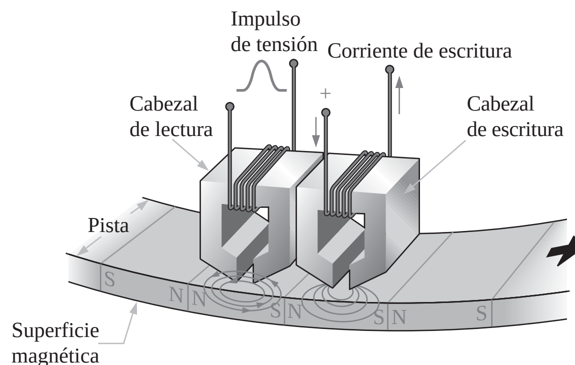
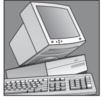


FIGURA 10.59 Funcionamiento simplificado del cabezal de lectura/escritura.

Formato del disco duro. Un disco duro está organizado en pistas y sectores, como muestra la Figura 10.60(a). Cada pista está dividida en una serie de sectores, y cada pista y sector tienen una dirección física que el sistema operativo utiliza para localizar un determinado registro de datos. Normalmente, los discos duros tienen desde unos pocos cientos hasta unos pocos miles de pistas. Como puede ver en la figura, hay un número constante de pistas por sector, utilizando los sectores externos una superficie mayor que los sectores internos. La disposición de pistas y sectores en un disco se denomina *formato.*

La Figura 10.60(b) muestra una pila de disco duro. Las unidades de disco duro difieren en el número de placas apiladas, aunque siempre hay un mínimo de dos. El conjunto de todas las pistas correspondientes de cada placa constituyen lo que se conoce colectivamente como cilindro, como se indica en la figura.



NOTAS INFORMÁTICAS

Los datos se almacenan en un disco duro en forma de archivos. Llevar la cuenta de la ubicación de los archivos es el trabajo del controlador de dispositivos que gestiona el disco duro (algunas veces denominado BIOS de la unidad de disco duro). El controlador de dispositivo y el sistema operativo de la computadora pueden acceder a dos tablas que se utilizar para controlar la ubicación y los nombres de los archivos. La primera tabla se denomina FAT (*File Allocation Table*, tabla de asignación de archivos). La FAT muestra qué partes se han asignado a archivos específicos y mantiene un registro de los sectores abiertos y los sectores defectuosos. La segunda tabla es el directorio raíz, que contiene el nombre de los archivos, el tipo de archivo, la fecha y hora de creación, el número del *cluster* inicial y otras informaciones referentes al archivo.

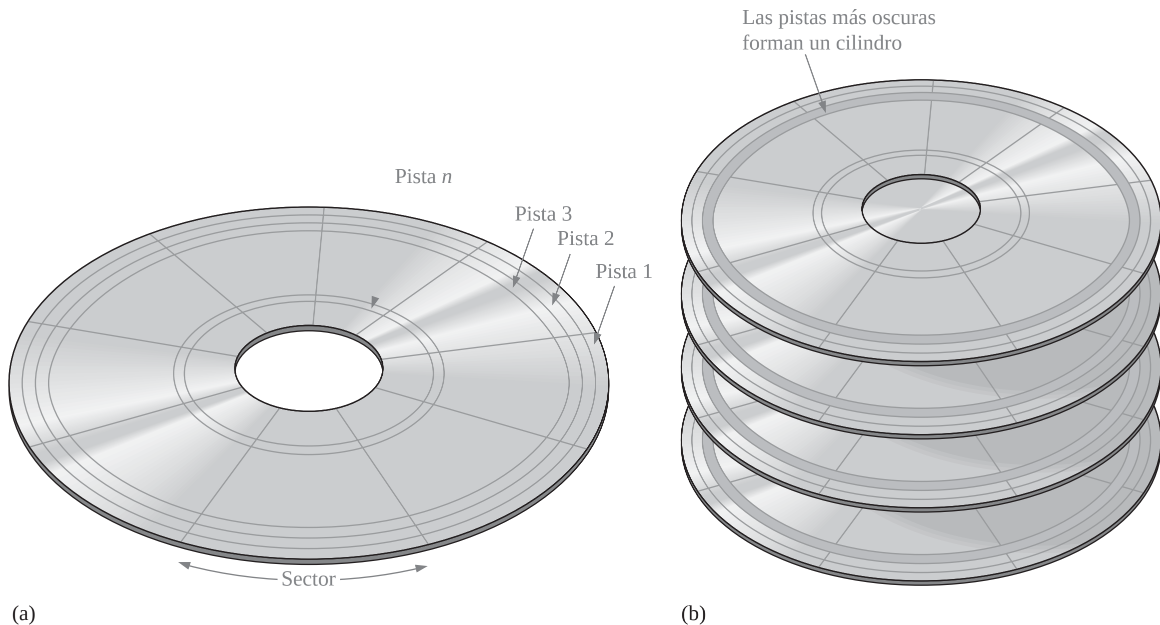


FIGURA 10.60 Organización y formato de un disco duro.

Funcionamiento del disco duro. El rendimiento de una unidad de disco duro específica viene determinado por varios parámetros básicos. Una operación de *búsqueda* consiste en el movimiento del cabezal de lectura-escritura hasta la pista deseada. El **tiempo de búsqueda** es el tiempo medio de realización de dicha operación. Normalmente, las unidades de disco duro tienen un tiempo medio de búsqueda de varios milisegundos, dependiendo de cada unidad concreta.

El **período de latencia** es el tiempo que tarda el sector deseado en colocarse debajo del cabezal, una vez que éste se ha posicionado en la pista deseada. El caso peor se produce cuando el sector deseado acaba de pasar la posición del cabezal y está girando, alejándose de la misma. El sector tiene que girar casi una revolución completa hasta alcanzar la posición del cabezal. El *período de latencia medio* supone que el disco debe recorrer media revolución. Obviamente, el período de latencia depende de la velocidad constante de rotación del disco. Las velocidades de rotación de disco son diferentes para distintas unidades de disco aunque, típicamente, son 3.600 rpm, 4.500 rpm, 5.400 rpm y 7.200 rpm. Algunas unidades de disco recientes giran a una velocidad de 10.033 rpm y tienen un período medio de latencia inferior a 3 ms.

La suma del tiempo medio de búsqueda y del período medio de latencia es el *tiempo de acceso* de la unidad de disco.

Discos flexibles. El nombre de disco flexible se debe a que este tipo de discos está hecho de un material de poliéster flexible, cubierto por ambas caras con una capa magnética. Los primeros discos flexibles tenían un diámetro de 5,25 pulgadas y estaban contenidos en una funda semiflexible. Los actuales **discos flexibles** o disquetes tienen un diámetro de 3,5 pulgadas y disponen de una funda de plástico rígido, como se muestra en la Figura 10.61. Una puerta con muelle cubre la ventana de acceso, y permanece cerrada hasta que el disquete se introduce en la unidad. El disco dispone de una placa metálica con un agujero para centrar el disquete y otro para hacerlo rotar dentro de la funda protectora. Obviamente, los disquetes son disquetes extraibles, mientras que los discos duros no. Los discos flexibles se formatean en pistas y sectores de forma similar a los discos duros, excepto por el número de pistas y sectores. Los disquetes de alta densidad de 1,44 MB tienen 80 pistas por cada lado, con 18 sectores.

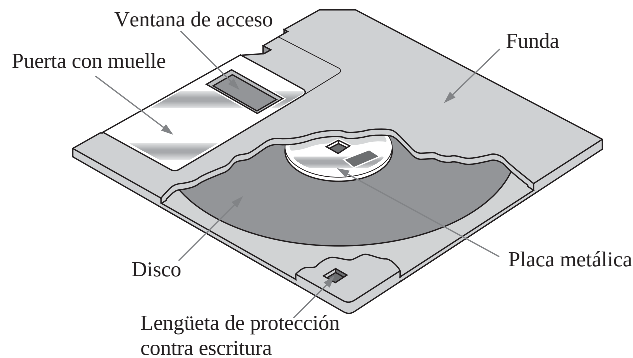


FIGURA 10.61 El disco flexible de 3,5 pulgadas (disquete).

Zip™. La unidad Zip es un tipo de dispositivo de almacenamiento magnético extraíble que parece ser el posible recambio de los disquetes de capacidad limitada. Al igual que el disco flexible, el cartucho del **disco Zip** es un disco flexible incluido en una carcasa rígida aproximadamente del mismo tamaño que el disquete, aunque más grueso. La unidad Zip es mucho más rápida que la unidad de disquetes, dado que tiene una velocidad de giro de 3.000 rpm frente a las 300 rpm de la unidad de disquete. La unidad Zip tiene una capacidad de almacenamiento de 250 MB, lo que es aproximadamente 173 veces mayor que la capacidad de 1,44 MB del disquete.

Jaz™. Otro tipo de dispositivo de almacenamiento magnético es la unidad Jaz, que es similar a una unidad de disco duro excepto en que las dos placas se encuentran dentro de un cartucho extraible protegido por un obturador a prueba de polvo. Los **cartuchos Jaz** están disponibles con capacidades de almacenamiento de 1 o 2 GB.

Disco duro extraible. Además de las populares unidades extraibles Zip y Jaz, hay disponibles discos duros extraibles con capacidades de 80 GB a 250 GB. Tenga presente que la tecnología está cambiando de forma tan rápida que probablemente ya habrá más avances en el momento en que se ponga a leer este libro.

Cinta magnética. La cinta se utiliza para realizar copias de seguridad de datos desde dispositivos de almacenamiento masivo y, normalmente, es mucho más lenta en términos de tiempo de acceso, ya que se accede a los datos en serie en lugar de mediante una selección aleatoria. Hay disponibles diversos tipos de cintas, entre los que se incluyen QIC, DAT, 8 mm y DLT.

QIC es la abreviatura de *quarter-inch cartridge* (cartucho de cuarto de pulgada) y se parece bastante a un casete de audio con dos carretes en el interior. Los distintos estándares QIC varían desde 36 a 72 pistas que pueden almacenar desde 80 MB hasta 1,2 GB. Las actualizaciones más recientes del estándar Travan han aumentado la longitud de la cinta y la anchura de la misma, permitiendo capacidades de almacenamiento de

hasta 4 GB. Las unidades de cinta QIC utilizan cabezales de lectura-escritura con un único cabezal de escritura y un cabezal de lectura en cada lado. Esto permite que la unidad de cinta compruebe los datos que se acaban de escribir, cuando la cinta está girando en cualquier dirección. En el modo de registro, la cinta se desplaza bajo los cabezales de lectura-escritura aproximadamente a una velocidad de 100 pulgadas por segundo (25,4 cm/s), como se indica en la Figura 10.62.

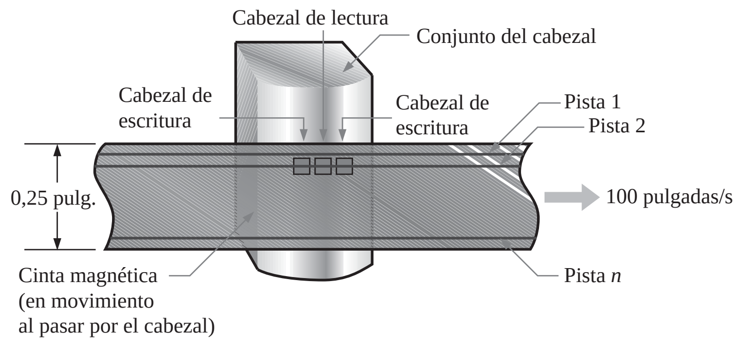


FIGURA 10.62 Cinta QIC.

DAT, que son las siglas de Digital Audio Tape (cinta digital de audio), utiliza una técnica denominada grabación por exploración helicoidal. Las cintas DAT ofrecen capacidades de almacenamiento en el rango de los 12 GB, pero son más caras que las QIC.

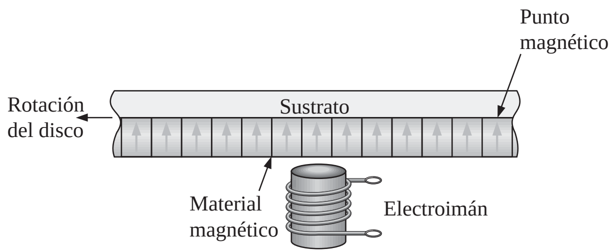
Un tercer tipo de formato de cinta es la cinta de 8 mm que, originalmente, se diseñó para la industria de vídeo, pero que ha sido adoptada por la industria informática como una forma fiable de almacenar grandes cantidades de datos de computadora. La cinta de 8 mm es similar a la DAT, pero ofrece capacidades de almacenamiento de hasta 25 GB.

DLT corresponde a las siglas de *Digital Linear Tape* (cinta digital lineal). La cinta DLT es una cinta de una anchura de media pulgada (1,25 cm), lo que significa que es un 60% más ancha que la cinta de 8 mm y, por tanto, dos veces más ancha que una cinta estándar QIC. Básicamente, la cinta DLT difiere en la forma en que trabaja el mecanismo de la unidad de cinta, para minimizar el deterioro de la cinta en comparación con otros sistemas. La cinta DLT ofrece la mayor capacidad de almacenamiento de todos los soportes físicos de este tipo, con capacidades que alcanzan hasta los 35 GB.

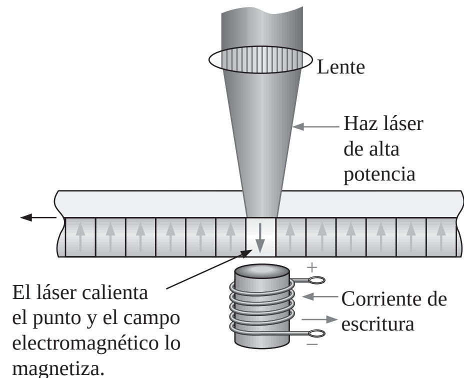
Almacenamiento magneto-óptico

Como su nombre indica, los dispositivos de almacenamiento magneto-óptico combinan las tecnologías magnética y óptica (láser). Un **disco magneto-óptico** se formatea en pistas y sectores de forma similar a los discos magnéticos.

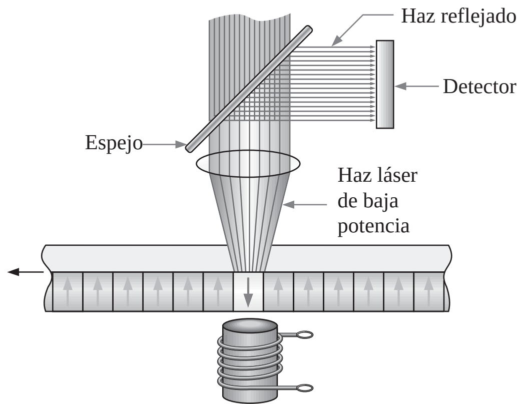
La diferencia básica entre un disco puramente magnético y un disco magneto-óptico es que la capa magnética utilizada en los discos magneto-ópticos requiere calor para alterar la polarización magnética. Por tanto, el disco magneto-óptico es extremadamente estable a temperatura ambiente, haciendo que los datos no cambien. Para escribir un bit de datos, se enfoca un haz láser de alta potencia sobre un punto muy pequeño del disco, y la temperatura de dicho punto se eleva por encima de un nivel de temperatura denominado punto de Curie (aproximadamente 200ºC). Una vez caliente, las partículas magnéticas en dicho punto pueden fácilmente ver cambiada su dirección (polarización) debido al efecto del campo magnético generado por el cabezal de escritura. La información se lee del disco mediante un láser de menor potencia que el que se emplea para escribir, utilizando el efecto de Kerr, según el cual la polaridad de la luz del láser reflejado se altera dependiendo de la orientación de las partículas magnéticas. Los puntos con cierta polaridad representan los ceros y los pun-



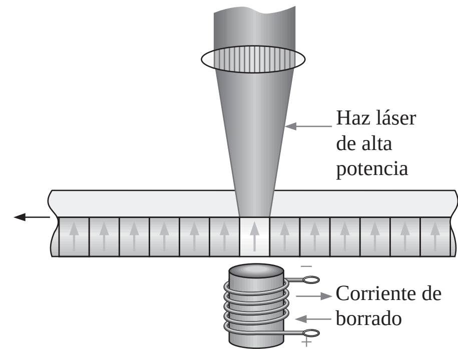
(a) Disco no grabado.



(b) Operación de escritura: un haz láser de alta potencia incide sobre un punto, haciendo que las partículas magnéticas se alineen con el campo electromagnético.



(c) Operación de lectura: un haz láser de baja potencia se refleja en las partículas magnéticas de polaridad inversa y su polarización cambia. Si las partículas no están invertidas, la polaridad del haz reflejado no cambia.



(d) Operación de borrado: el campo electromagnético se invierte al mismo tiempo que el haz láser de alta potencia incide en un punto, haciendo que las partículas magnéticas vuelvan a la polaridad original.

FIGURA 10.63 Principios básicos de un disco magneto-óptico.

tos con la polaridad opuesta representan los unos. El funcionamiento básico magneto-óptico se muestra en la Figura 10.63, que representa una pequeña sección transversal de un disco.

Almacenamiento óptico

CD-ROM. La memoria de sólo lectura de disco compacto (*compact-disc read-only memory*) es un disco de 120 mm de diámetro con tres capas dispuestas en forma de sandwich: la capa inferior de plástico de policarbonato, una hoja delgada de aluminio para la reflectividad y una capa superior de laca para protección. El disco **CD-ROM** se formatea con una única pista en forma de espiral, con sectores secuenciales de 2 kB y tiene una capacidad de 680 MB. Los datos se pregraban en fábrica en forma de agujeros microscópicos denominados *muescas* y el área plana que rodea a estos agujeros se denomina *planicie.* Las muescas se imprimen en la capa de plástico y no pueden borrarse.

Un reproductor de CD-ROM lee los datos en la pista espiral mediante un haz láser de infrarrojos de baja potencia, como se muestra en la Figura 10.64. Los datos están codificados mediante las muescas y planicies,

como muestra la figura. La luz del láser reflejada desde una muesca tiene un desfase de 180º respecto de la luz reflejada desde las planicies. Cuando el disco gira, el estrecho haz de láser incide sobre las muescas y planicies de longitudes variables, y un fotodiodo detecta la diferencia en la luz reflejada. El resultado es una serie de unos y ceros, que corresponde a la configuración de las muescas y de las planicies a lo largo de la pista.

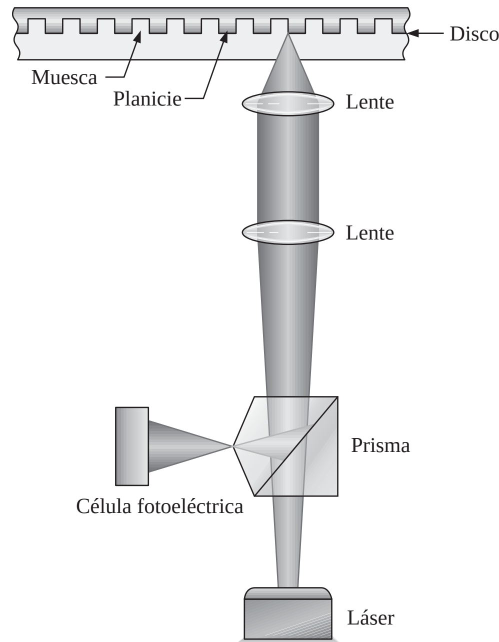


FIGURA 10.64 Operación básica de lectura de datos en un CD-ROM.

WORM. Los discos de única escritura- múltiples lecturas (*write once -read many*) son un tipo de dispositivo de almacenamiento óptico en los que se puede escribir una sola vez, después de lo cual no se pueden borrar los datos, aunque sí se pueden leer muchas veces. Para escribir los datos se utiliza un haz láser de baja potencia, que crea agujeros microscópicos en la superficie del disco. Los unos y los ceros quedan representados por las áreas en las que hay y no hay agujero.

CD-R. Prácticamente, es un tipo de WORM. La diferencia se encuentra en que el CD grabable permite múltiples sesiones de escritura en diferentes áreas del disco. El disco CD-R tiene una pista en forma de espiral como el CD-ROM, pero, en lugar de hacerse agujeros mecánicamente sobre el disco para representar los datos, el CD-R emplea un láser para quemar agujeros micróscopicos en una superficie con tinte orgánico. Cuando se calienta más allá de una temperatura crítica con un haz láser durante la operación de lectura, los puntos quemados cambian de color y reflejan menos luz que las áreas no quemadas. Por tanto, los unos y los ceros se representan en un CD-R mediante las áreas quemadas y no quemadas, mientras que un CD-ROM se representan mediante las muescas y las planicies. Al igual que con el CD-ROM, los datos no pueden borrarse una vez que se han escrito.

CD-RW. El disco CD regrabable puede utilizarse para leer y escribir datos. En lugar de la capa de grabación con tinte que se emplea en el CD-R, el CD-RW normalmente utiliza un compuesto cristalino con una propiedad especial. Cuando se calienta a una cierta temperatura, al enfriarse se vuelve cristalino, pero si se calienta a una temperatura superior, se funde y se vuelve amorfo al enfriarse. Para escribir datos, el haz láser enfocado calienta el material a la temperatura de fundido dando lugar al estado amorfo. Las áreas amorfas resultan-

tes reflejan menos la luz que las áreas cristalinas, permitiendo que la operación de lectura detecte los unos y los ceros. Los datos se pueden borrar o sobrescribir calentado las áreas amorfas a una temperatura superior a la temperatura de cristalización, pero inferior a la temperatura de fusión, lo que hace que el material amorfo pase de nuevo al estado cristalino.

DVD-ROM. Originalmente, DVD eran las siglas correspondientes a Digital Video Disk (videodisco digital) pero, actualmente, corresponden a *Digital Versatile Disk*. Al igual que en el CD-ROM, en el **DVD-ROM** los datos se pregraban en el disco. Sin embargo, el tamaño de las muescas es menor que en el CD-ROM, lo que permite almacenar más datos en una pista. La diferencia principal entre el CD-ROM y el DVD-ROM es que el CD-ROM tiene una única cara mientras que el DVD almacena datos por las dos caras. También, además de los discos DVD de dos caras, hay disponibles discos de múltiples capas que utilizan capas de datos semitransparentes colocadas sobre las capas de datos principales, proporcionando capacidades de almacenamiento de decenas de gigabytes. Para acceder a todas las capas, hay que cambiar el enfoque del haz láser para pasar de una capa a otra.

REVISIÓN DE LA SECCIÓN 10.8

1. Enumerar los principales tipos de dispositivos de almacenamiento magnético.
2. ¿Cuál es actualmente la capacidad de almacenamiento de los discos flexibles?
3. Generalmente, ¿cómo está organizado un disco magnético?
4. ¿Cómo se escriben y se leen los datos en un disco magneto-óptico?
5. Enumerar los tipos de almacenamiento óptico.

10.9 LOCALIZACIÓN DE AVERÍAS

Ya que las memorias pueden contener una gran cantidad de celdas de almacenamiento, comprobar cada una de ellas puede ser un proceso muy largo y frustrante. Afortunadamente, las memorias usualmente se prueban mediante un procedimiento automático realizado con un equipo de pruebas programable, o con la ayuda de un software para comprobación interna del sistema. La mayoría de los sistemas basados en microprocesador proporcionan la comprobación automática de memoria como parte de su software de sistema.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Explicar el método de suma de comprobación utilizado para verificar memorias ROM. ■ Explicar el método del patrón ajedrezado utilizado para probar memorias RAM.

Comprobación de una ROM

Puesto que las ROM contienen datos conocidos, se puede comprobar la corrección de los datos almacenados leyendo cada palabra de datos de la memoria, y comparándola con la palabra de datos que se sabe que es correcta. En la Figura 10.65 se ilustra una forma de hacer esto. Este procedimiento requiere una ROM de referencia que contenga los mismos datos que la ROM que se va a comprobar. Un equipo de pruebas especial se programa para leer cada dirección de ambas memorias ROM simultáneamente y comparar los contenidos. El organigrama de la Figura 10.66 presenta la secuencia básica.

Método de la suma de comprobación. Aunque el método anterior comprueba cada dirección de la ROM para asegurar la corrección de los datos, tiene la desventaja de que se requiere una ROM de referencia para cada ROM diferente que se desee probar. También puede ocurrir que la ROM de referencia falle, dando lugar a una indicación falsa de error.

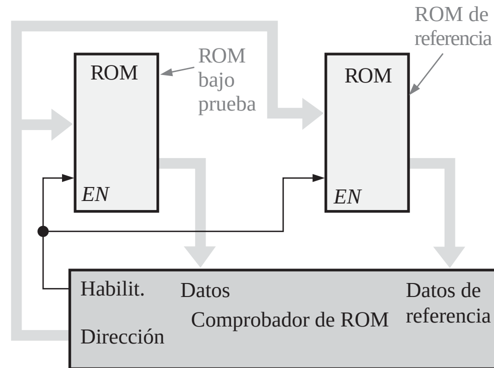


FIGURA 10.65 Diagrama de bloques para la comprobación de todo el contenido de una ROM.

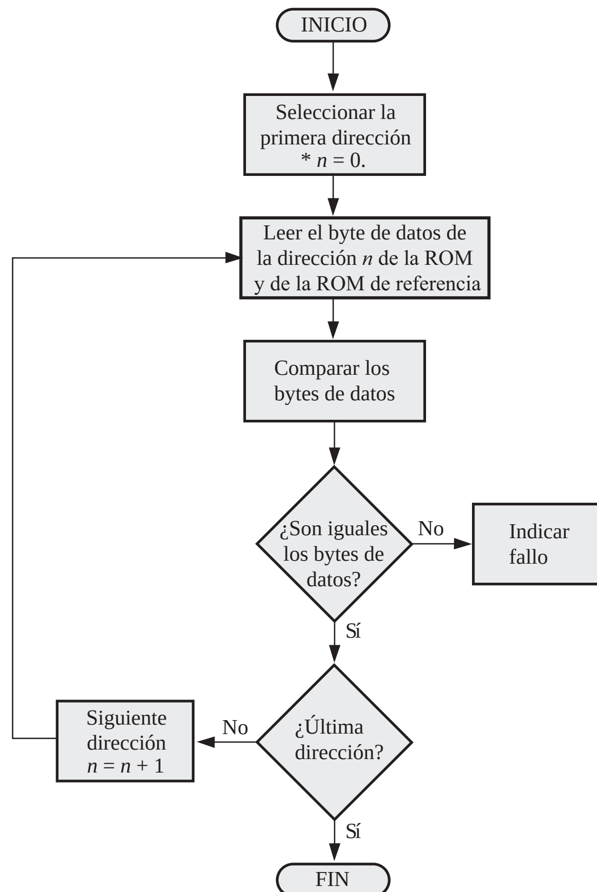


FIGURA 10.66 Organigrama para la comprobación de todo el contenido de una ROM.

En el método de suma de comprobación, un número, que es la suma de todos los contenidos de todas las direcciones de la ROM, se almacena en una dirección especifica de la memoria cuando ésta se programa. Para

probar la ROM, se suman los contenidos de todas las direcciones, excepto la propia suma de verificación, y el resultado se compara con la suma de verificación almacenada en la ROM. Si existe diferencia, es seguro que hay un fallo. Si las sumas de verificación son iguales, muy probablemente la ROM está bien. Sin embargo, existe la remota posibilidad de que una combinación de celdas de memoria erróneas haga que las sumas de verificación sean iguales.

En la Figura 10.67 se ilustra este procedimiento mediante un sencillo ejemplo. En este caso, la suma de comprobación se genera sumando cada columna de bits de datos y descartando los acarreos. Esto equivale a realizar la operación XOR en cada columna. El organigrama de la Figura 10.68 presenta el método de prueba básico mediante la suma de comprobación.

Este método de prueba puede implementarse con un equipo de pruebas especial, o puede incorporarse en una rutina de prueba dentro del software del sistema, en los sistemas basados en microprocesador. En este caso, la rutina de prueba de la ROM se ejecuta automáticamente al arrancar el sistema.

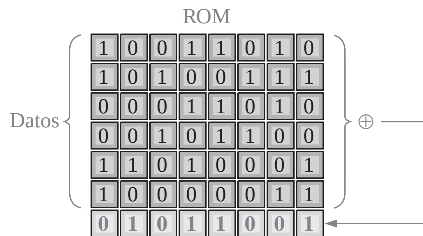


FIGURA 10.67 Ilustración simplificada de una ROM programada, con una suma de comprobación almacenada en una dirección específica.

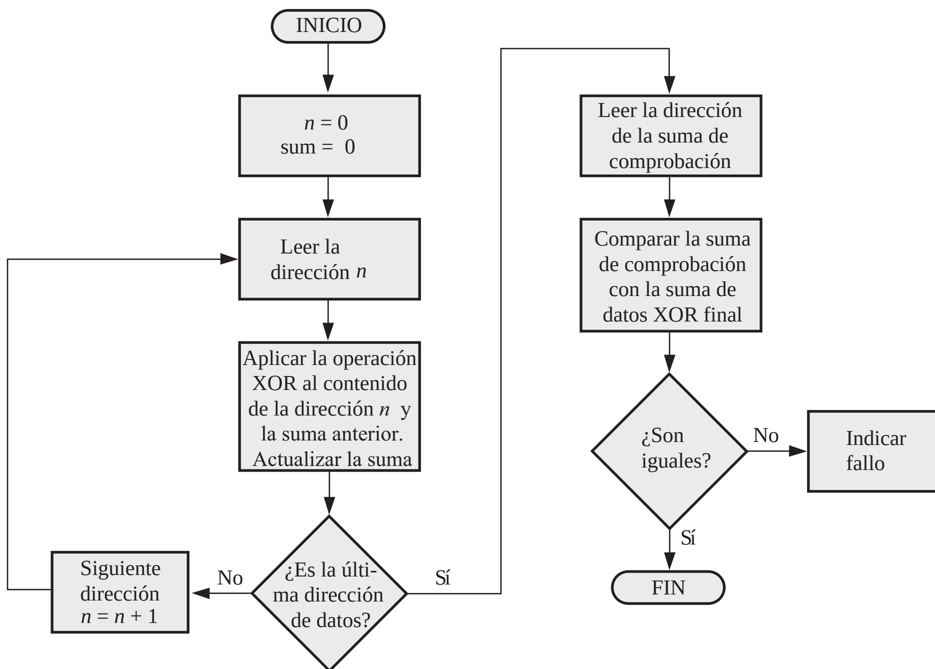


FIGURA 10.68 Organigrama para el método básico de prueba mediante la suma de comprobación.

Comprobación de la RAM

Para probar la capacidad de las memorias RAM de almacenar 0s y 1s en sus celdas se escriben, en primer lugar 0s en todas las celdas de cada dirección, y luego se extraen y verifican. A continuación, se escriben 1s en todas las celdas de cada dirección, y después se leen y verifican. Este método de pruebas básico detectará una celda que se mantenga en el estado 1 o en el estado 0.

Algunos fallos de memoria no se pueden detectar escribiendo ceros en todas las direcciones y unos en todas las direcciones. Por ejemplo, si dos celdas de memoria adyacentes se cortocircuitan, siempre estarán en el mismo estado, siendo ambas 0, o ambas 1. También este método de prueba es inefectivo si existen problemas de ruido interno, consistentes en que los contenidos de una o más direcciones se alteran debido al cambio de los contenidos de otras direcciones.

Método de prueba con patrón ajedrezado. Una forma más completa para probar una RAM consiste en utilizar un patrón de 1s y 0s alternativos, como ilustra la Figura 10.69. Observe que todas las celdas adyacentes contienen bits opuestos. Este patrón comprueba que no haya un cortocircuito entre dos celdas adyacentes, ya que, si existe un cortocircuito, ambas células estarán en el mismo estado.

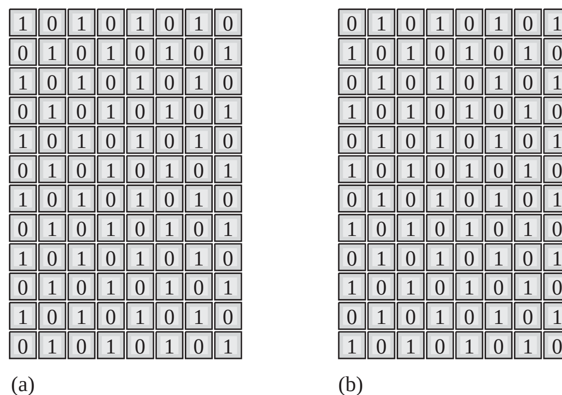


FIGURA 10.69 Patrón de prueba ajedrezado de una RAM.

Después de comprobar la RAM con el patrón de la Figura 10.69(a), éste se invierte, como se muestra en la parte (b) de la figura. Esta inversión verifica la capacidad de todas las celdas de almacenar tanto un 1 como un 0.

Una comprobación más consiste en alternar el patrón en una dirección de memoria cada vez y comprobar que el patrón original no se haya modificado en las restantes direcciones. Esta prueba detectará los problemas consistentes en que los contenidos de una dirección se alteren dinámicamente cuando los contenidos de otra dirección cambien.

En el organigrama de la Figura 10.70 se presenta un procedimiento básico para la prueba del patrón ajedrezado. El procedimiento se puede implementar con el software del sistema en los sistemas basados en microprocesador, de modo que las comprobaciones se realizan automáticamente cuando se enciende el sistema o se pueden iniciar desde el teclado.

REVISIÓN DE LA SECCIÓN 10.9

1. Describir el método de suma de comprobación para probar una ROM.
2. ¿Por qué no se puede aplicar el método de suma de comprobación para probar una RAM?
3. Enumerar los tres fallos básicos que se pueden detectar en una RAM con la prueba del patrón ajedrezado.

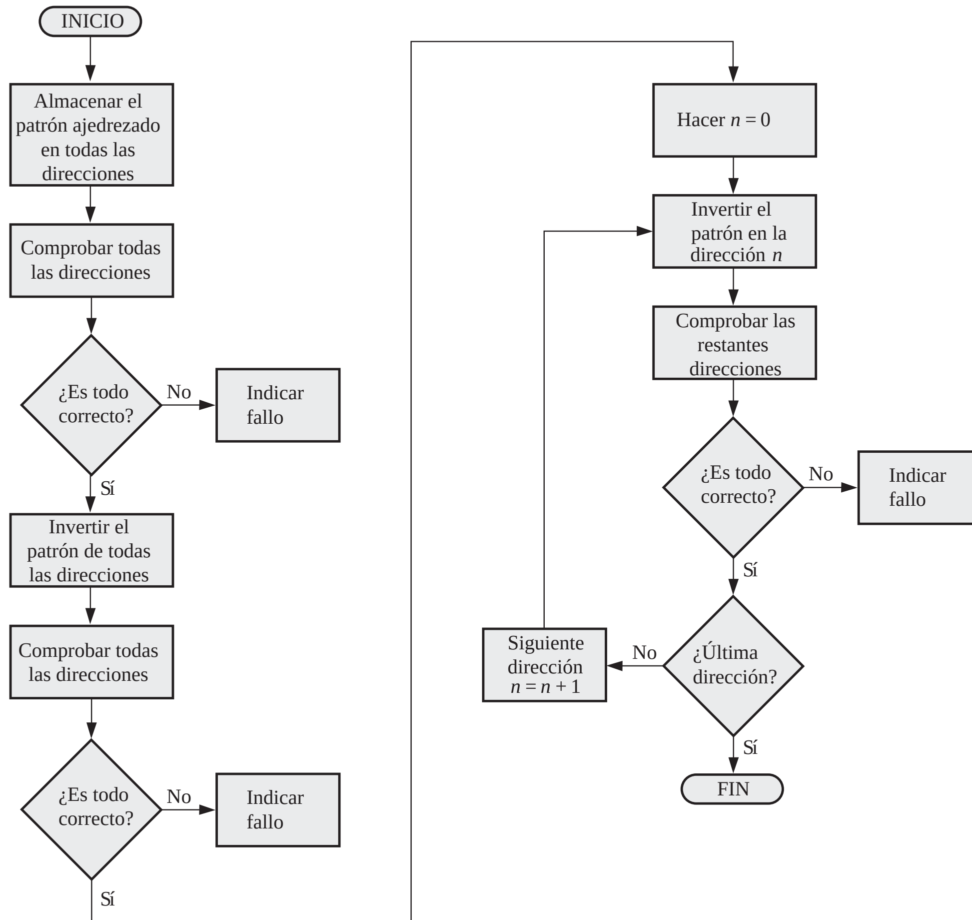


FIGURA 10.70 Organigrama para el método básico de prueba del patrón ajedrezado.



APLICACIÓN A LOS SISTEMAS DIGITALES

En esta aplicación se desarrolla la lógica de memoria para el sistema de seguridad presentado en el Capítulo 9. En el capítulo anterior se ha completado la lógica para el código de seguridad, la cual se combinará con la lógica de memoria para formar el sistema completo.

Funcionamiento general

En la Figura 10.71 se presenta un diagrama de bloques básico del sistema completo de seguridad de acceso.

La lógica de memoria almacena un código de acceso de 4 dígitos en formato BCD. En el modo de desactivación (*Desarm*), se introducen en la memoria cuatro dígitos a través del teclado. Una vez almacenados en la memoria, los cuatro dígitos BCD pasan a ser el código de seguridad permanente que hay que introducir. Si fuera necesario cambiar el código de seguridad, la memoria se reprogramaría con uno nuevo.

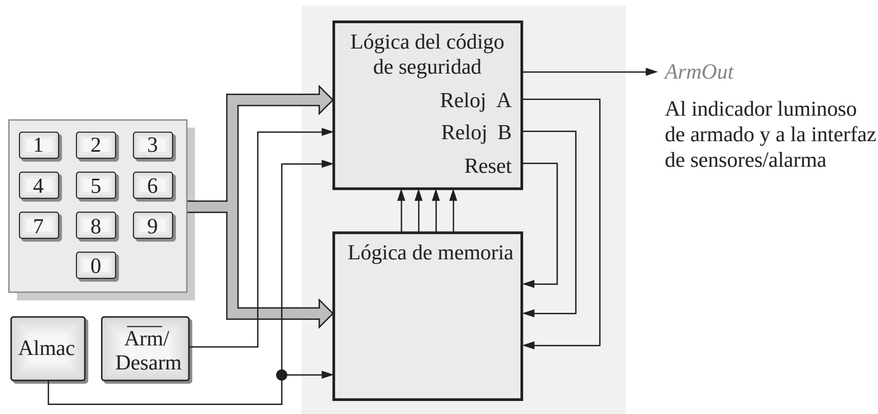


FIGURA 10.71 Diagrama de bloques básico del sistema de seguridad de acceso.

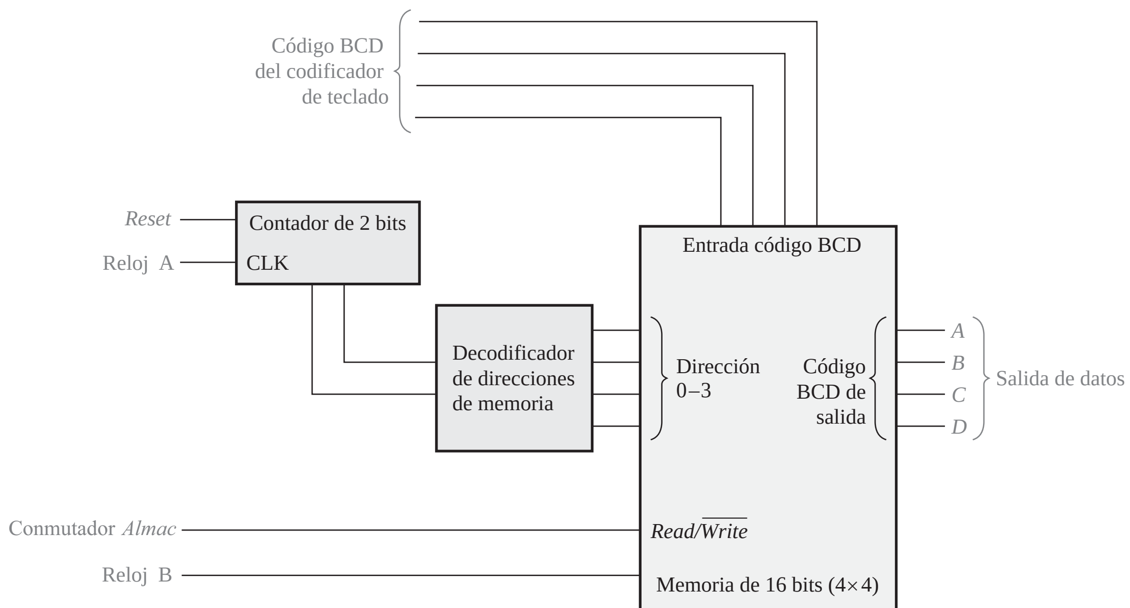


FIGURA 10.72 Diagrama de bloques de la lógica de memoria.

La memoria se programa poniendo en primer lugar el sistema en modo de desactivación (*desarm)* y utilizando el conmutador *Almacenar* y el teclado para introducir el código de cuatro dígitos deseado. Esto es una operación de *escritura* en memoria. Una vez que se ha programado la memoria con el código de seguridad, el conmutador $\overline{\text{Arm}}$/Desarm se cambia al modo *arm* (activación), que prepara a la memoria para realizar operaciones de *lectura.* En la Figura 10.72 se muestra un diagrama de bloques de la lógica de memoria.

La celda de memoria

La memoria requiere 16 celdas para almacenar los cuatro dígitos BCD del código de seguridad. En la Figura 10.73 se muestra un posible diseño de una celda de memoria. Se utiliza un flip-flop J-K como dispositivo elemento básico de almacenamiento; dicho flip-flop puede operar en dos modos (*lectura* y *escritura*). En el modo de *escritura*, *SelDir* (selección de dirección) está a nivel ALTO y la entrada $R/\overline{W}$ (*read/ write*, *lectura / escritura*) está a nivel BAJO. Las puertas *G*1 y *G*2 están habilitadas, el bit de entrada se aplica a la entrada *J* y su complemento se apli-

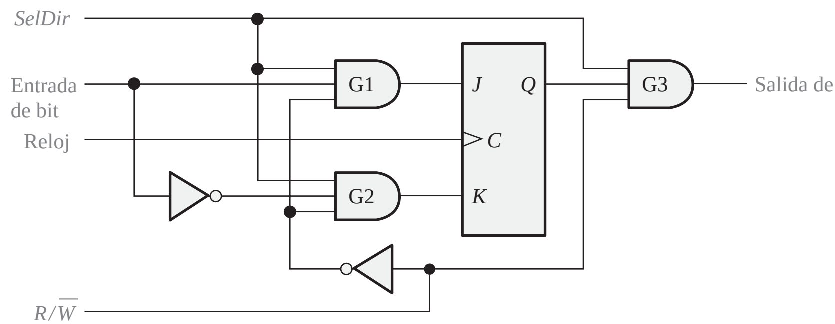


FIGURA 10.73 Lógica de la celda de memoria.

ca a la entrada *K*. El bit de entrada se almacena entonces con el flanco positivo del impulso de reloj. En el modo de *lectura*, *SelDir* está a nivel ALTO y $R/\overline{W}$ está también a nivel ALTO, habilitando *G*3. El bit almacenado en la salida *Q* del flip-flop aparece en la salida de *G*3 (Salida de bit).

El decodificador de direcciones de memoria

En la Figura 10.74 se presenta la lógica del decodificador de direcciones de memoria. Se aplica una secuencia binaria de 2 bits a las entradas de selección (S_0, S_1) para seleccionar cada una de las cuatro direcciones de memoria utilizando las líneas *SelDir*.

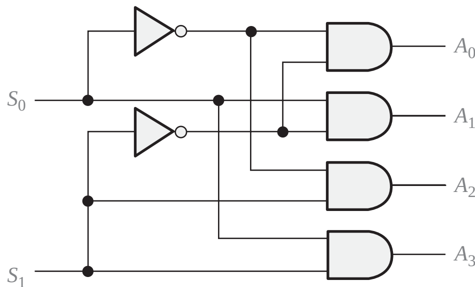


FIGURA 10.74 Decodificador de direcciones de memoria.

La matriz de memoria

La memoria tiene dieciseis celdas como se muestra en la Figura 10.75. Cuando el decodificador de direcciones selecciona una de las filas de memoria y la entrada $\overline{read/write}$ está a nivel BAJO, el código de entrada BCD de 4 bits se almacena a impulsos del reloj en las cuatro celdas seleccionadas. Las entradas al decodificador de direcciones pasan secuencialmente a través de cada uno de los cuatro estados (00, 01, 10 y 11) para seleccionar sucesivamente cada fila de la memoria.

La Figura 10.76 ilustra la programación de la memoria a medida que se introduce secuencialmente el código de seguridad 4739.

Lógica de memoria completa

Es necesario un codificador de teclado para convertir una pulsación de tecla en un código BCD, y se utiliza un contador de 2 bits para producir la secuencia de selección de las direcciones de memoria. Esto se muestra en la Figura 10.77. Al comienzo de la programación, se reinicializa el contador al estado 0 mediante una entrada de reinicialización procedente de la lógica de introducción del código, y el contador avanza a través de la secuencia de introducción de cada dígito.

Sistema de seguridad completo

Ahora que hemos completado la lógica de memoria, podemos combinarla con la lógica del código de seguridad del Capítulo 9 como muestra el diagrama de bloques de la Figura 10.78, para formar el sistema de seguridad completo mostrado como diagrama de bloques en la Figura 10.79.

Práctica de sistemas

- ***Actividad 1.*** Explicar qué función tiene el codificador de teclado en la lógica de memoria.
- ***Actividad 2.*** Explicar qué función tiene el contador de 2 bits en la lógica de memoria.
- ***Actividad opcional.*** Construir el sistema de seguridad de acceso completo usando dispositivos estándar 74XX y otros componentes que sean necesarios. Probar el sistema.

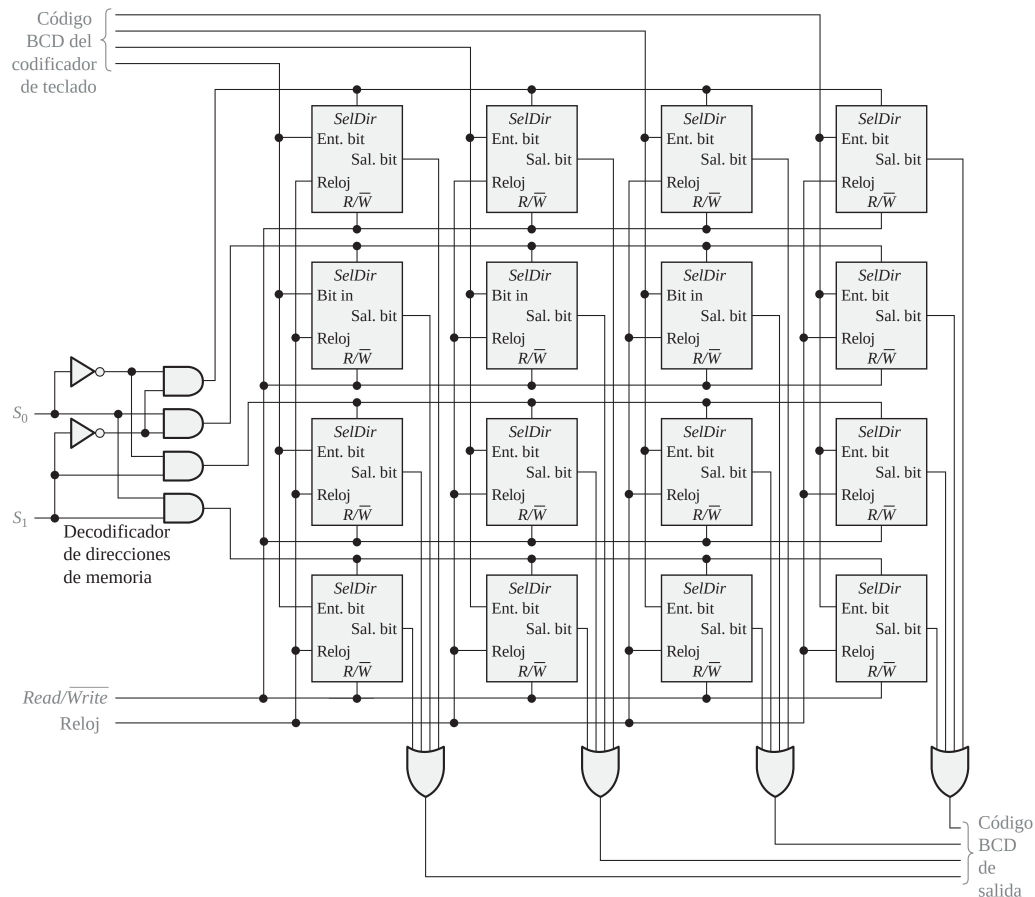


FIGURA 10.75 Matriz de memoria y decodificador de direcciones.

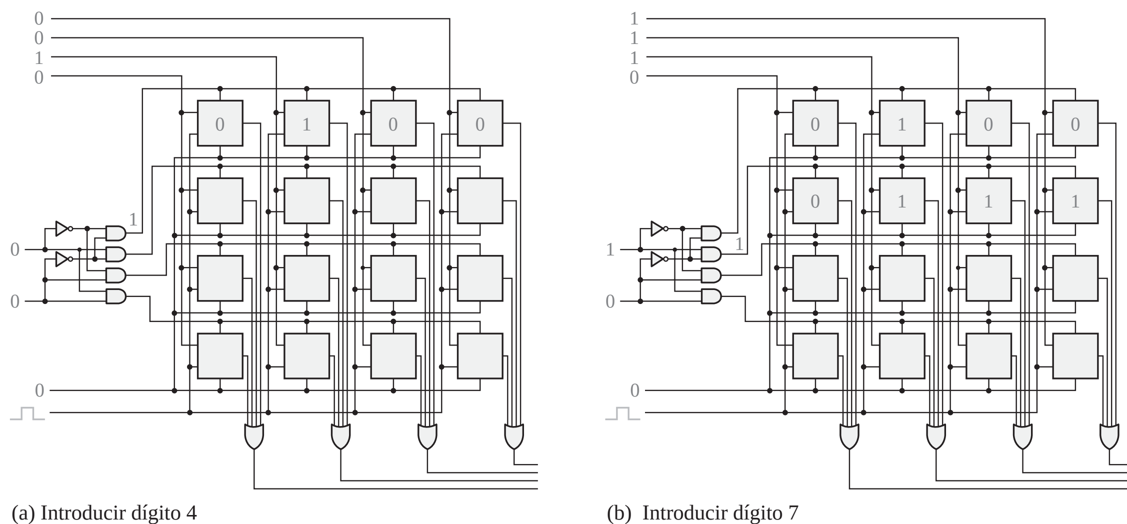
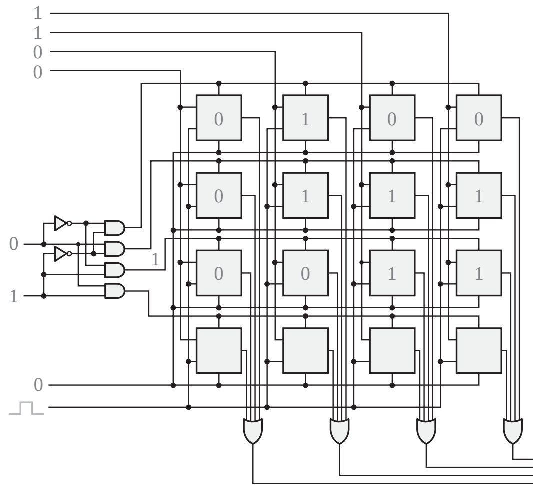
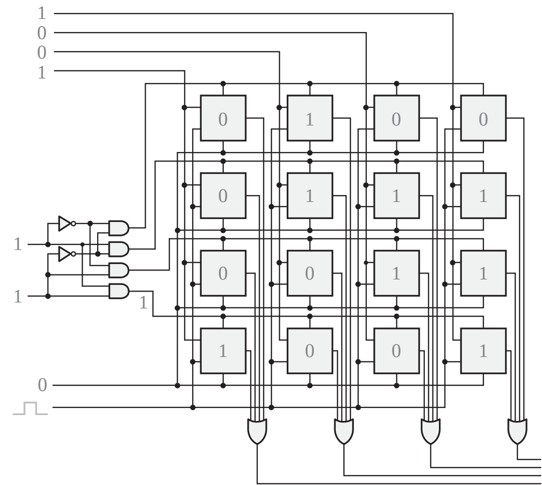


FIGURA 10.76 Ilustración de cómo se introduce en memoria un código de seguridad (4739). (*Continúa*)



(c) Introducir dígito 3



(d) Introducir dígito 9

FIGURA 10.76 Ilustración de cómo se introduce en memoria un código de seguridad (4739). (*Continuación*)

Código BCD
del codificador
de teclado

Reset
Reloj A
Contador
de 2 bits
CLK

Read/$\overline{Write}$
(Conmutador *Almac*)
Reloj B

A
B
C
D
Salida de datos

FIGURA 10.77 Lógica de memoria completa.

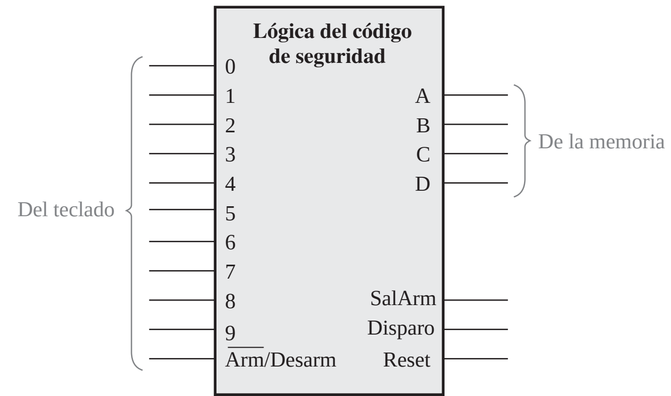


FIGURA 10.78 Lógica del código de seguridad (del Capítulo 9).

Lógica de memoria

Del teclado

0
1
2
3
4
5
6
7
8
9

SalDatos0
SalDatos1
SaliDatos2
SalDatos3

Almac
Reloj

Disparo
Almac
Reset
Entrada de reloj

Lógica del código de seguridad

0
1
2
3
4
5
6
7
8
9
$\overline{\text{Arm}}$/Desarm

A
B
C
D

SalArm
Disparo
Reset

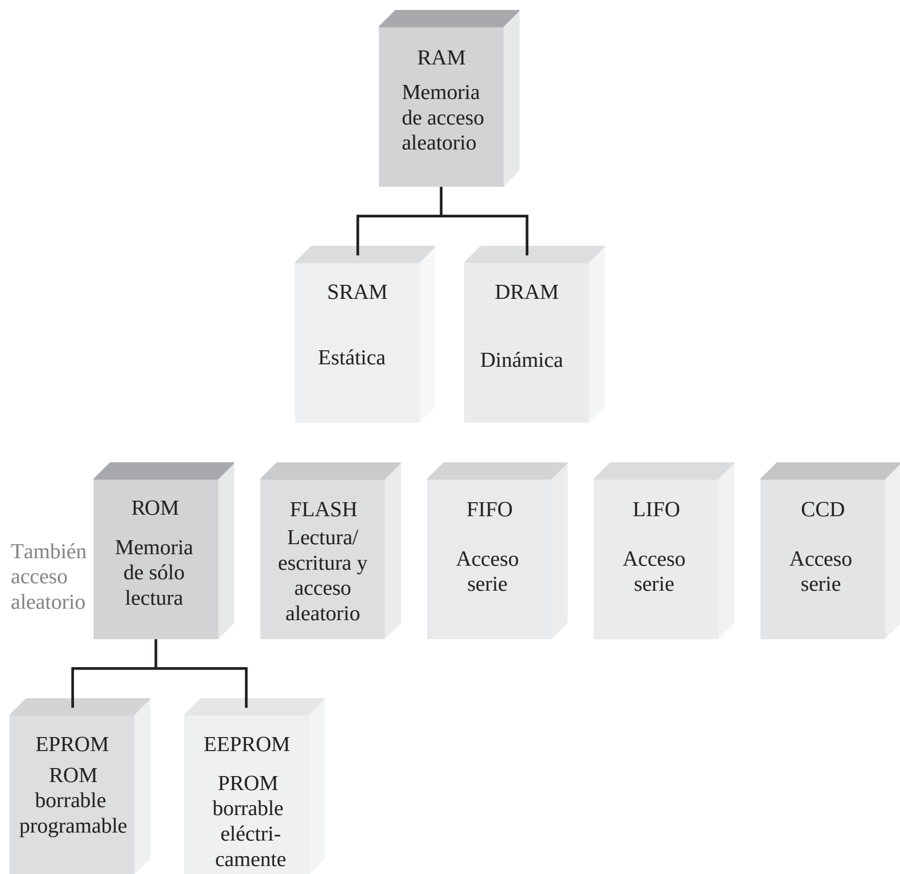
SalArm

$\overline{\text{Arm}}$/Desarm

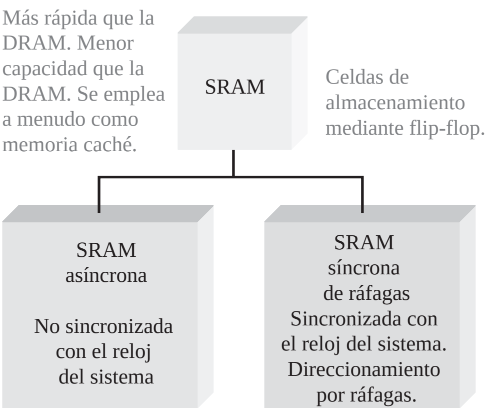
FIGURA 10.79 El sistema de seguridad completo.

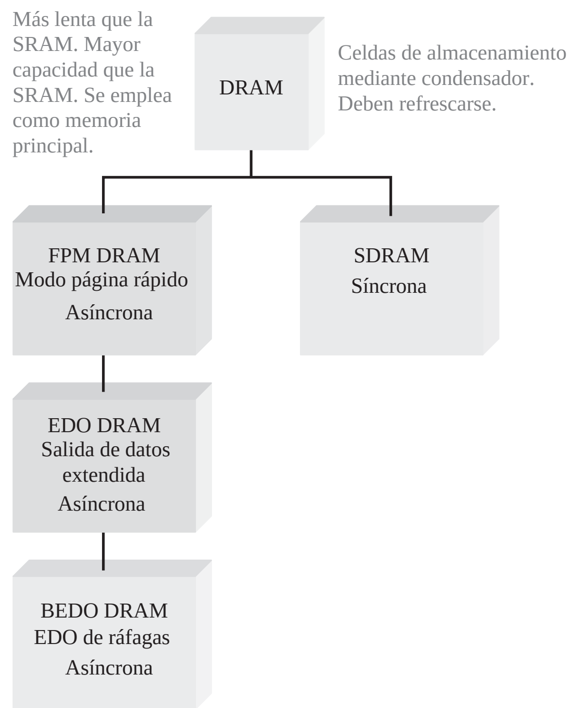
RESUMEN

- Tipos de memorias semiconductoras:

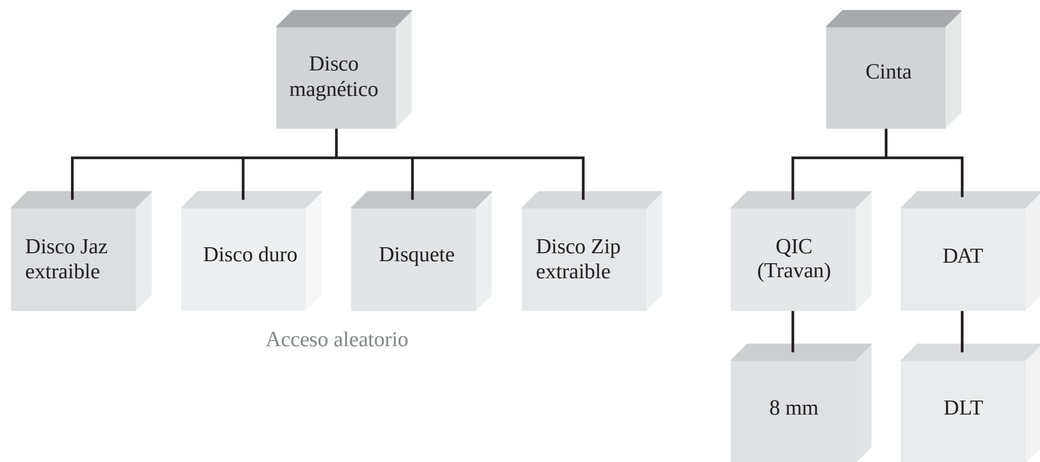


- Tipos de memorias SRAM (RAM estática) y DRAM (RAM dinámica):

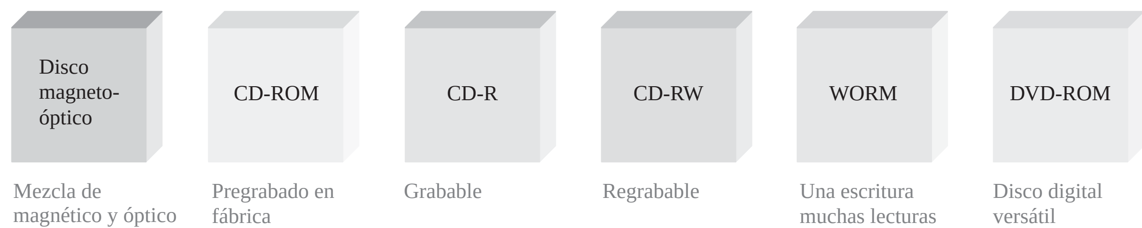




■ Tipos de dispositivos de almacenamiento magnético



■ Tipos de dispositivos de almacenamiento óptico (láser):



PALABRAS CLAVE

Las palabras clave y otros términos que se han resaltado en negrita se encuentran en el glosario final del libro.

Bus Conjunto de interconexiones que establece la interfaz entre uno o más dispositivos basándose en una especificación estandarizada.
Byte Grupo de ocho bits.
Capacidad Número total de unidades de datos (bits, nibbles, bytes, palabras) que puede almacenar una memoria.
Celda Un elemento de almacenamiento en una memoria.
Dirección Posición de una determinada celda de almacenamiento o grupo de celdas en memoria.
Disco duro Dispositivo de almacenamiento magnético; normalmente, una pila de dos o más discos rígidos encerrados en un compartimento sellado.
DRAM *Dynamic Random-Access Memory*, memoria dinámica de acceso aleatorio. Un tipo de memoria semiconductora que utiliza condensadores como elemento de almacenamiento y es una memoria de lectura/escritura volátil.
EPROM *Electrically Erasable Programmable Read-Only Memory*, un tipo de memoria semiconductora que normalmente utiliza luz ultravioleta para borrar los datos.
Escritura El proceso de almacenar datos en memoria.
FIFO *First In-First Out*, primero en entrar-primero en salir.
Lectura El proceso de recuperar datos de una memoria.
LIFO *Last In-First Out*, memoria de tipo primero en entrar-último en salir. Pila de memoria.
Memoria flash Memoria semiconductora de acceso aleatorio de lectura/escritura no volátil, en la que los datos se almacenan como carga en la puerta flotante de un determinado tipo de FET.
Palabra Unidad completa de datos binarios.
PROM *Programmable Read-Only Memory* (memoria de sólo lectura programable); un tipo de memoria semiconductora.
RAM *Random-Access Memory*, memoria de acceso aleatorio. Memorias semiconductoras volátiles de lectura/escritura.
ROM *Read-Only Memory*, memoria semiconductora no volátil de acceso aleatorio.
SRAM *Static Random Access Memory*, memoria estática de acceso aleatorio; un tipo de memoria volátil semiconductora de lectura/escritura.

AUTOTEST

Las respuestas se encuentran al final del capítulo.

1. La capacidad de bits de una memoria que tiene 1024 direcciones y que puede almacenar 8 bits en cada dirección es:

(a) 1024 **(b)** 8192 **(c)** 8 **(d)** 4096

2. Una palabra de datos de 32 bits está formada por:

(a) 2 bytes **(b)** 4 nibbles **(c)** 4 bytes **(d)** 3 bytes y 1 *nibble*

3. Los datos en una memoria de acceso aleatorio (RAM) se almacenan durante:

(a) la operación de lectura.

(b) la operación de habilitación.

(c) la operación de escritura.

(d) la operación de direccionamiento.

4. Los datos que se almacenan en una determinada dirección de una memoria de acceso aleatorio (RAM) se pierden cuando:

(a) se apaga la alimentación.

(b) se leen los datos de dicha dirección.

(c) se escriben nuevos datos en dicha dirección.
(d) las respuestas (a) y (c).

5. Una ROM es:
(a) una memoria no volátil.
(b) una memoria volátil.
(c) una memoria de lectura/escritura.
(d) una memoria organizada en bytes.

6. Una memoria con 256 direcciones tiene:
(a) 256 líneas de dirección.
(b) 6 líneas de dirección.
(c) 1 línea de dirección.
(d) 8 líneas de dirección.

7. Una memoria organizada en bytes tiene:
(a) 1 línea de salida de datos.
(b) 4 líneas de salida de datos.
(c) 8 líneas de salida de datos.
(d) 16 líneas de salida de datos.

8. La celda de almacenamiento en una SRAM es:
(a) un flip-flop **(b)** un condensador
(c) un fusible **(d)** un punto magnético

9. Una DRAM debe ser:
(a) reemplazada periódicamente.
(b) refrescada periódicamente.
(c) habilitada siempre.
(d) programada antes de cada uso.

10. Una memoria flash es:
(a) volátil
(b) una memoria de sólo lectura.
(c) una memoria de lectura/escritura.
(d) no volátil.
(e) las respuestas (a) y (c).
(f) las respuestas (c) y (d).

11. Disco duro, disquete, disco Zip y disco Jaz son todos ellos:
(a) dispositivos de almacenamiento magneto-óptico.
(b) dispositivos de almacenamiento semiconductores.
(c) dispositivos de almacenamiento magnéticos.
(d) dispositivos de almacenamiento ópticos.

12. Los dispositivos de almacenamiento óptico emplean:
(a) luz ultravioleta.
(b) campos electromagnéticos.
(c) acopladores ópticos.
(d) láseres.

PROBLEMAS

Las respuestas a los problemas impares se encuentran al final del libro.

SECCIÓN 10.1 Principios de las memorias semiconductoras

1. Identificar la ROM y la RAM de la Figura 10.80.

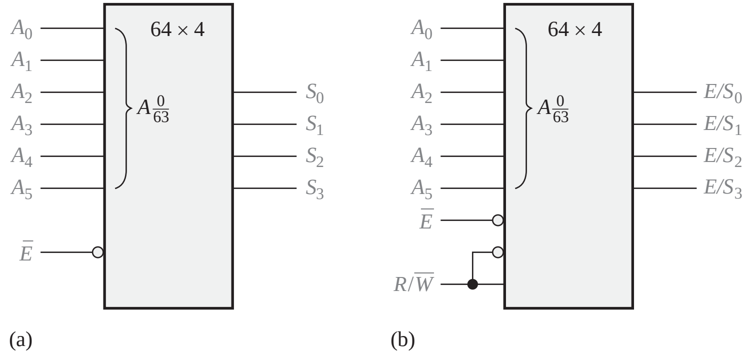


FIGURA 10.80

2. Explicar por qué las ROM y las RAM son memorias de acceso aleatorio.

3. Explicar los propósitos del bus de direcciones y del bus de datos.

4. Cuál es la dirección de memoria (de 0 hasta 256) que representa cada uno de los siguientes números hexadecimales:

(a) $0A_{16}$ **(b)** $3F_{16}$ **(c)** CD_{16}

SECCIÓN 10.2 Memorias de acceso aleatorio (RAM)

5. En una matriz de memoria estática con cuatro filas similar a la de la Figura 10.9, se almacenan inicialmente todo ceros. ¿Cuál es el contenido después de las siguientes condiciones? Suponer que un 1 selecciona una fila.

Fila 0 = 1, Entrada de datos (bit 0) = 1,

Fila 1 = 0, Entrada de datos (bit 1) = 1,

Fila 2 = 1, Entrada de datos (bit 2) = 1,

Fila 3 = 0, Entrada de datos (bit 3) = 0

6. Dibujar un diagrama lógico básico para una RAM estática de 512 × 8 bits, indicando todas las entradas y salidas.

7. Suponiendo que una SRAM de 64k × 8 tiene una estructura similar a la SRAM de la Figura 10.11, determinar el número de filas y de columnas de 8 bits en la matriz de celdas de memoria.

8. Dibujar de nuevo el diagrama de bloques de la Figura 10.11 para una memoria de 64k × 8.

9. Explicar la diferencia entre una SRAM y una DRAM.

10. ¿Cuál es la capacidad de una DRAM con doce líneas de dirección?

SECCIÓN 10.3 Memorias de sólo lectura (ROM)

11. Para la matriz ROM de la Figura 10.81, determinar las salidas para todas las posibles combinaciones de entrada, y resumirlas en forma de tabla (celda en gris claro es 1, celda en gris oscuro es 0).

12. Determinar la tabla de verdad de la ROM de la Figura 10.82.

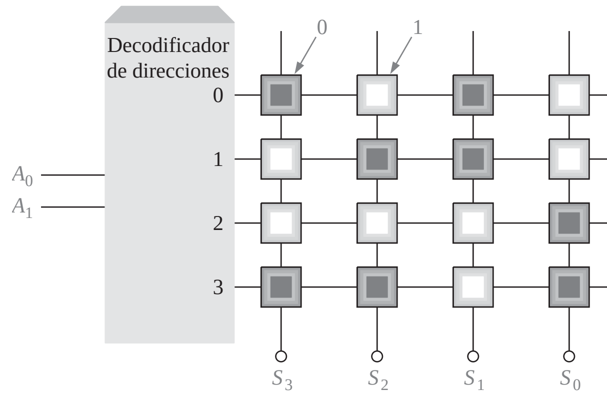


FIGURA 10.81

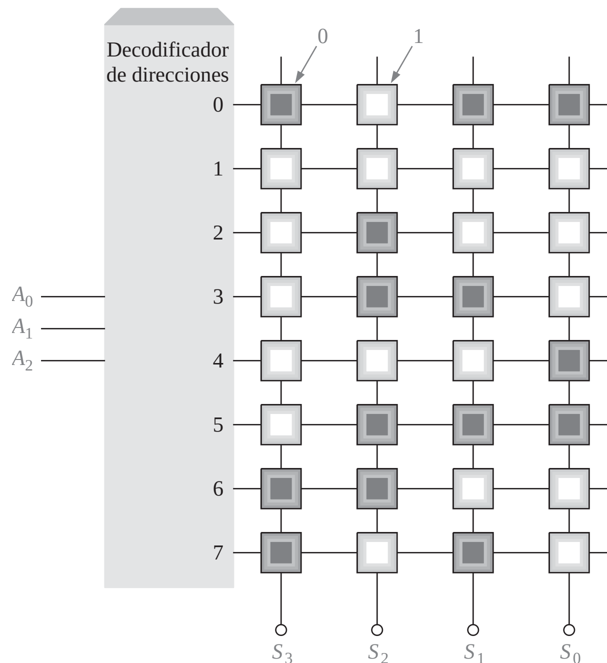


FIGURA 10.82

13. Utilizando un procedimiento similar al del Ejemplo 10.1, diseñar una ROM para convertir un único dígito BCD en código de exceso 3.

14. ¿Cuál es la capacidad total de bits de una ROM que tiene 14 líneas de dirección y 8 salidas de datos?

SECCIÓN 10.4 Memorias ROM programables (PROM y EPROM)

15. Suponer que la matriz PROM de la Figura 10.83 se programa fundiendo un hilo fusible para crear un 0. Indicar los hilos que hay que fundir para programar una tabla índice para la operación X^3, donde X es un número de 0 a 7.

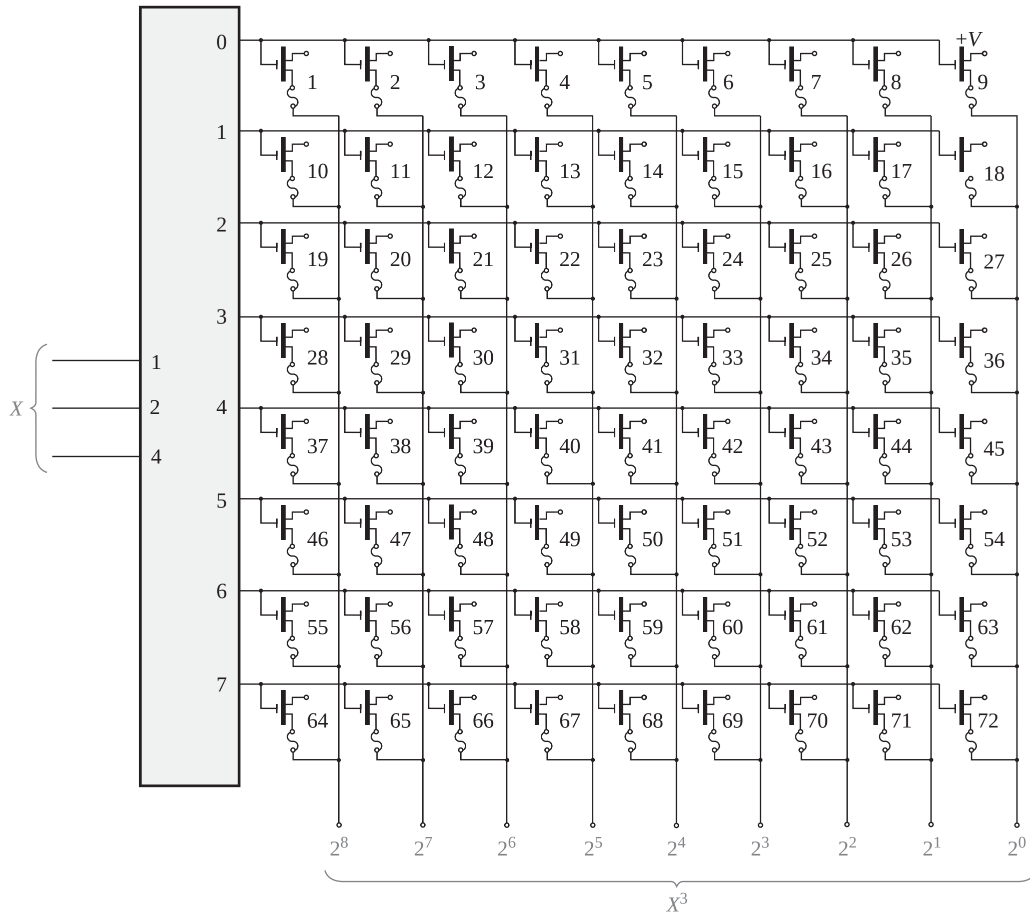


FIGURA 10.83

16. Determinar las direcciones que se programan y los contenidos de cada dirección después de aplicar la secuencia de programación de la Figura 10.84 a una EPROM como la mostrada en la Figura 10.31.

SECCIÓN 10.6 Expansión de memoria

17. Utilizar memorias DRAM de 16k × 4 para formar una RAM de 64k × 8. Dibujar el diagrama lógico.

18. Utilizando un diagrama de bloques, demostrar cómo se pueden expandir memorias RAM dinámicas de 64k × 1 para formar una RAM de 256k × 4.

19. ¿Cuál es la longitud de palabra y la capacidad de palabra de la memoria del Problema 17? ¿Y del Problema 18?

SECCIÓN 10.7 Tipos especiales de memorias

20. Completar el diagrama de tiempos de la Figura 10.85, mostrando las formas de onda de salida, que inicialmente están a nivel BAJO, para una memoria serie FIFO como la mostrada en la Figura 10.49.

21. Considerar una RAM de 4096 × 8 en la que las 64 últimas direcciones se usan como pila LIFO. Si la primera dirección de la RAM es 000_{16}, indicar las 64 direcciones utilizadas para la pila.

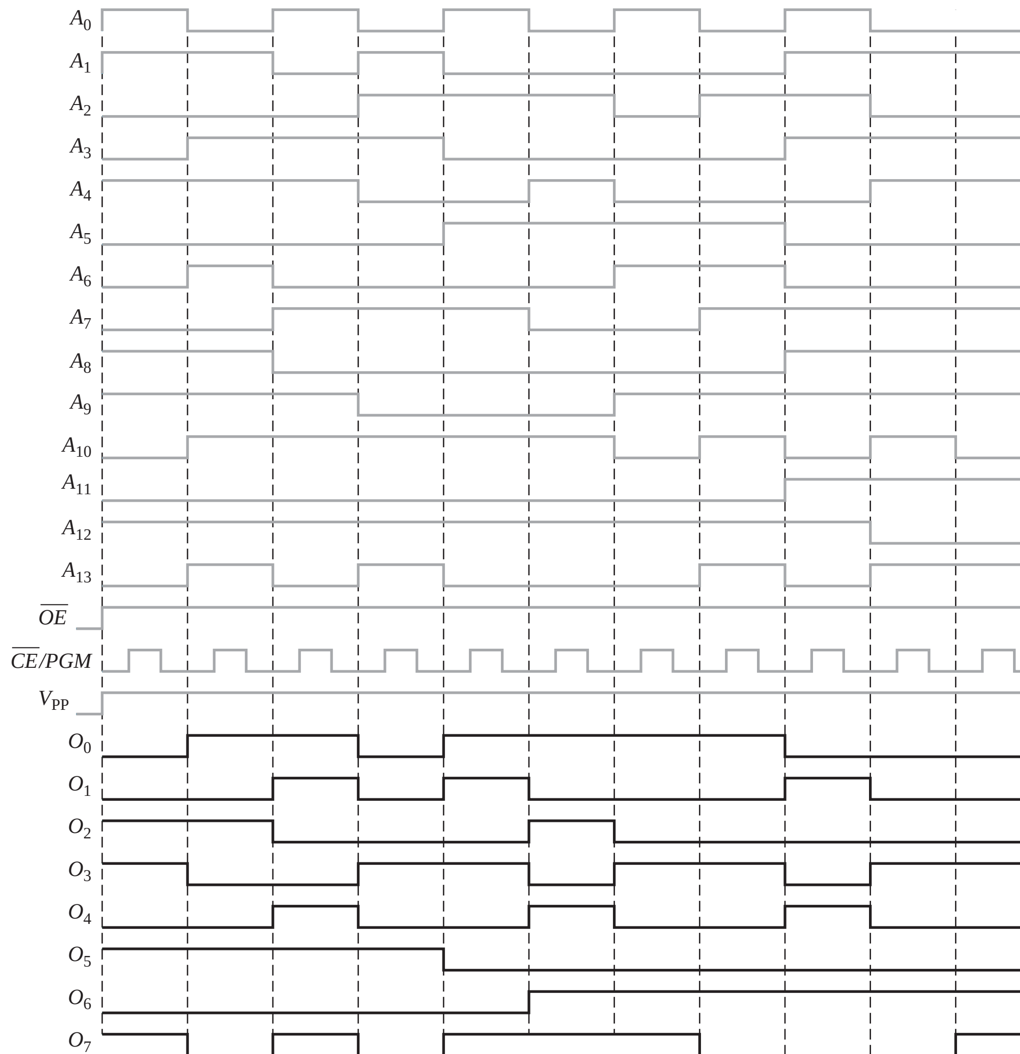


FIGURA 10.84

22. En la memoria del Problema 21, se introducen 16 bytes en la pila. ¿En qué dirección se encuentra el primer byte? ¿Y el último byte?

SECCIÓN 10.8 Dispositivos de almacenamiento magnético y óptico

23. Describir el formato general de un disco duro.

24. Explicar qué es el tiempo de búsqueda y el período de latencia en una unidad de disco duro.

25. ¿Por qué la cinta magnética requiere un tiempo de acceso mucho mayor que un disco?

26. Explicar las diferencias entre un disco magneto-óptico, un CD-ROM y un WORM.

SECCIÓN 10.9 Localización de averías

27. Determinar si los contenidos de la ROM de la Figura 10.86 son correctos.

28. Una ROM de 128k × 8 se implementa como se indica en la Figura 10.87. El decodificador decodifica los dos bits de dirección más significativos para habilitar una de las ROM cada vez, dependiendo de la dirección seleccionada.

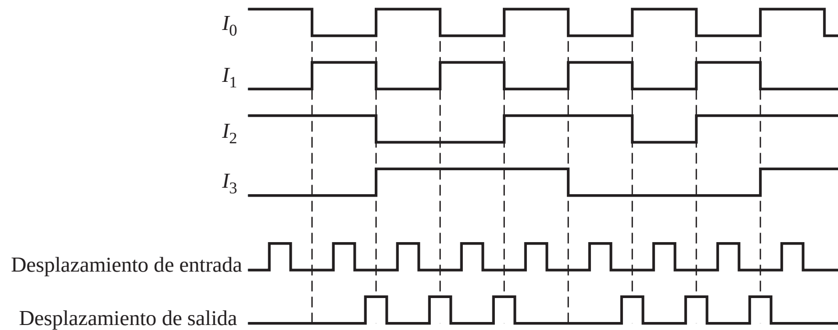


FIGURA 10.85

	ROM
	1 0 1 1 1
	1 1 1 1 0
	1 1 0 1 1
	1 0 1 1 0
	1 1 1 0 1
	1 1 1 0 0
	0 0 0 0 1
Suma de comprobación	0 1 1 0 0

FIGURA 10.86

(a) Expresar la dirección más baja y la más alta de cada ROM con números hexadecimales.

(b) Suponer que se usa una única suma de comprobación para la memoria completa y se almacena en la dirección más alta. Desarrollar el organigrama para probar el sistema de memoria completo.

(c) Suponer que cada ROM tiene una suma de comprobación almacenada en su dirección más alta. Modificar el organigrama desarrollado en el apartado (b) para reflejar este cambio.

(d) ¿Cuál es la desventaja de utilizar una única suma de comprobación para la memoria completa en lugar de una suma de comprobación para cada ROM individual?

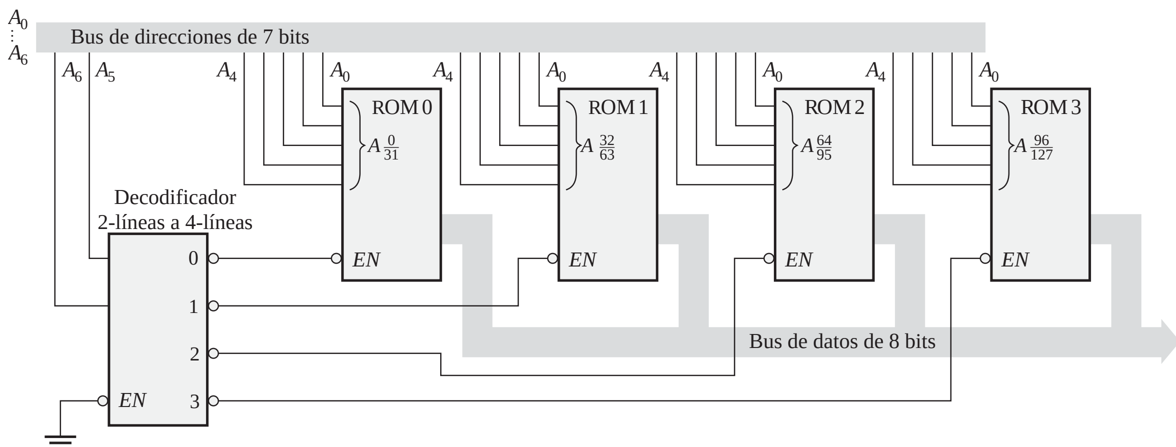


FIGURA 10.87

29. Suponer que se ejecuta una prueba de suma de comprobación en la memoria de la Figura 10.87 y que cada ROM tiene su suma de comprobación en la dirección más alta. ¿Cuál o cuáles circuitos integrados reemplazaría para cada uno de los siguientes mensajes de error que aparecen en el monitor de vídeo del sistema?

(a) DIRECCIÓN 40 - 5F FALLO

(b) DIRECCIÓN 20 - 3F FALLO

(c) DIRECCIÓN 00 - 7F FALLO



Aplicación a los sistemas digitales

30. Desarrollar un cronograma de la lógica de memoria básica de la Figura 10.72, para ilustrar la introducción de los dígitos 4321 en la RAM. Incluir todas las entradas y salidas de los dispositivos.

31. En el modo de programación, ¿cuál es el estado del contador de la Figura 10.77 después de haber introducido dos dígitos del código?

32. ¿Cuál es el propósito de la lógica de memoria?

33. Comentar las ventajas y desventajas de utilizar una PROM externa al CPLD en lugar de la memoria del chip CPLD en la lógica de memoria.



Problemas especiales de diseño

34. Modificar el diseño de la lógica de memoria del sistema de seguridad de acceso, para acomodar un código de acceso de 5 dígitos.

37. Realizar las modificaciones apropiadas en la lógica de introducción del código en el sistema de seguridad de acceso, para un código de acceso de 5 dígitos. Consulte la sección "Aplicación a los sistemas digitales" del Capítulo 9.

RESPUESTAS

SECCIÓN 10 Principios de las memorias semiconductoras

1. El bit es la unidad de datos más pequeña.

2. 2048 bits son 256 bytes

3. Una operación de escritura almacena datos en la memoria.

4. Una operación de lectura realiza una copia de datos de la memoria.

5. Una unidad de datos se localiza mediante una dirección

6. Una RAM es volátil y tiene capacidad de lectura/escritura. Una ROM es no volátil y sólo tiene capacidad de lectura.

SECCIÓN 10.2 Memorias de acceso aleatorio (RAM)

1. Ráfaga síncrona y asíncrona.

2. Una pequeña memoria rápida entre la UCP y la memoria principal.

3. Las SRAM tienen celdas de almacenamiento *latch* que pueden mantener indefinidamente los datos. Las DRAM tienen celdas de almacenamiento capacitivo que se deben refrescar periódicamente.

4. La operación de refresco evita que los datos se pierdan debido a la descarga del condensador. Cada bit almacenado se restaura periódicamente recargando el condensador a su nivel nominal.

5. FPM, EDO, BEDO, síncrona

SECCIÓN 10.3 **Memorias de sólo lectura (ROM)**

1. 512 × 8 es igual a 4096 bits.
2. ROM de máscara, PROM, EPROM, UV EPROM, EEPROM.
3. Se requieren ocho bits de dirección para 256 localizaciones de bytes ($2^8 = 256$).

SECCIÓN 10.4 **Memorias ROM programables (PROM y EPROM)**

1. Las PROM son programables eléctricamente, las ROM no.
2. Los bits quedan a 1 después de borrar la EPROM.
3. La lectura es el modo normal de operación de una PROM.

SECCIÓN 10.5 **Memorias flash**

1. Flash, ROM, EPROM y EEPROM son no volátiles.
2. La memoria flash es no volátil. Las memorias SRAM y DRAM son volátiles.
3. Programación, lectura, borrado.

SECCIÓN 10.6 **Expansión de memoria**

1. Ocho memorias RAM.
2. Ocho memorias RAM.
3. SIMM: módulo de memoria de una única fila de pines.
4. DIMM: módulo de memoria con doble fila de pines.
5. RIMM: *rambus in-line memory module*.

SECCIÓN 10.7 **Tipos especiales de memoria**

1. En una memoria FIFO el *primer* bit (o palabra) *en entrar* es el *primer* bit (o palabra) *en salir*.
2. En una memoria LIFO el *último* bit (o palabra) *en entrar* es el *primer* bit (o palabra) *en salir*. Una pila es una LIFO.
3. La operación o instrucción que añade datos a la pila de memoria.
4. La operación o instrucción que elimina datos de la pila de memoria.
5. CCD significa dispositivo de acoplamiento de carga.

SECCIÓN 10.8 **Dispositivos de almacenamiento magnético y óptico**

1. Almacenamiento magnético: disquetes, disco duro, cinta y disco magneto-óptico.
2. Capacidad de almacenamiento de los discos flexibles: 1,44 MB.
3. Un disco magnético está organizado en pistas y sectores.
4. Un disco magneto-óptico utiliza un rayo láser y un electroimán.
5. Almacenamiento óptico: CD-ROM, CD-R, CD-RW, DVD-ROM, WORM.

SECCIÓN 10.9 **Localización de averías**

1. Los contenidos de la ROM se suman y comparan con una suma de comprobación prealmacenada.
2. La suma de comprobación no se puede utilizar porque los contenidos de una RAM no son fijos.
3. (1) un cortocircuito entre celdas adyacentes. (2) la incapacidad de algunas celdas para almacenar tanto 1s como 0s. (3) la alteración dinámica de los contenidos de una dirección cuando los contenidos de otra varían.

PROBLEMAS RELACIONADOS

10.1 $G_3G_2G_1G_0 = 1110$

10.2 Deben conectarse ocho memorias ROM de 64k × 1 en paralelo para formar una ROM de 64k × 8.

10.3 Dieciséis memorias ROM de 64k × 1.

10.4 *Véase* la Figura 10.88.

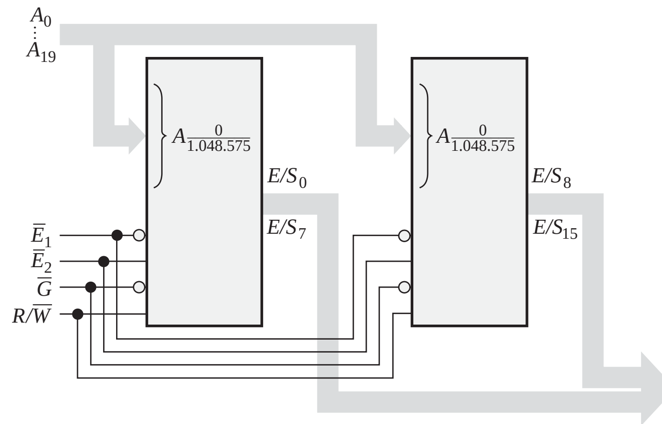


FIGURA 10.88

10.5 ROM 1: 0 a 524.287; ROM 2: 524.288 a 1.048.575

AUTOTEST

1. (b) **2.** (c) **3.** (c) **4.** (d) **5.** (a) **6.** (d) **7.** (c)
8. (a) **9.** (b) **10.** (f) **11.** (c) **12.** (d)

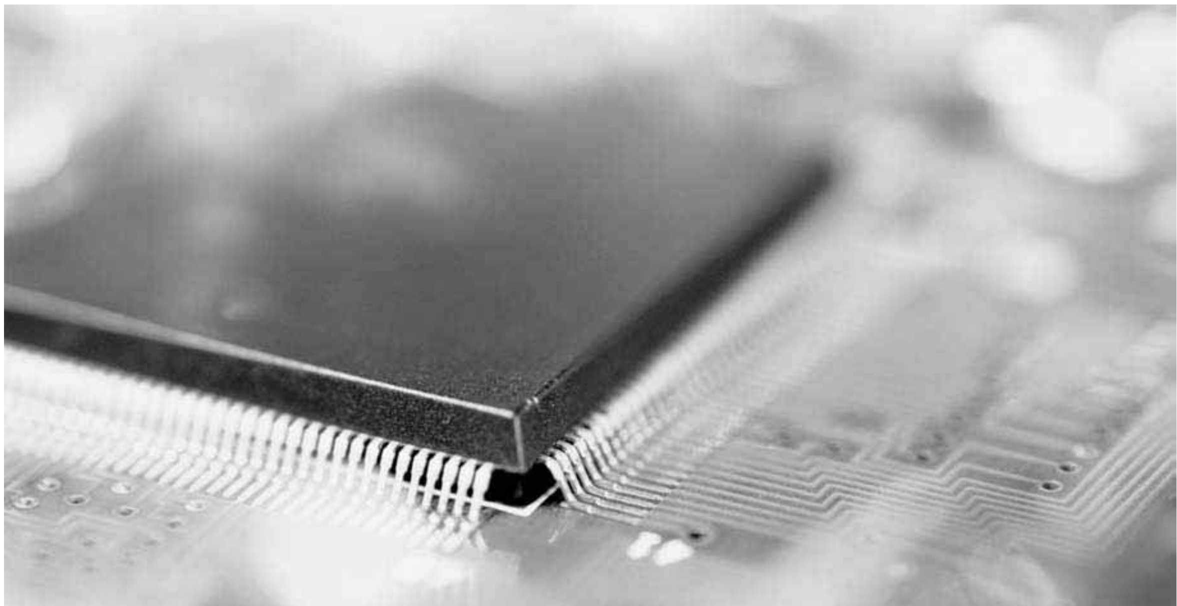
11

SOFTWARE Y LÓGICA PROGRAMABLE

CONTENIDO DEL CAPÍTULO

OBJETIVOS DEL CAPÍTULO

- Explicar los tipos de lógica programable, dispositivos SPLD y CPLD, así como su estructura básica.
- Describir la arquitectura básica de los dos tipos de dispositivos SPLD: la PAL y la GAL.
- Describir la arquitectura de la familia MAX 7000 de Altera de dispositivos CPLD.
- Describir la arquitectura de la familia de los dispositivos CPLD MAX II de Altera.
- Explicar la estructura básica de una matriz lógica programable (PLA, *programmable logic array*).



- Describir la arquitectura de la familia de CoolRunner II de dispositivos CPLD de Xilinx.
- Explicar el funcionamiento de las macroceldas.
- Distinguir entre dispositivos CPLD y FPGA.
- Explicar el funcionamiento básico de una tabla de consulta LUT (*look-up table*).
- Definir *propiedad intelectual* y *FPGA de plataforma.*
- Describir la arquitectura de la familia Stratix de dispositivos FPGA de Altera.
- Explicar las funciones integradas.
- Describir la arquitectura de la familia Virtex de FPGA de Xilinx.
- Mostrar un flujo básico de diseño software para dispositivos programables.
- Explicar los elementos del flujo de diseño correspondientes a la introducción del diseño, la simulación funcional, la síntesis, la implementación, la simulación de temporización y la descarga.
- Exponer varios métodos de prueba de dispositivos lógicos programables incluyendo la lógica de exploración de contorno.

PALABRAS CLAVE

- PAL
- GAL
- Macrocelda
- Registrado
- CPLD
- LAB
- LUT
- FPGA
- CLB
- Propiedad intelectual
- Flujo de diseño
- Dispositivo objetivo
- Introducción de esquemáticos
- Introducción de texto
- Compilador
- Descarga
- Cama de pinchos
- Sonda volante
- Exploración de contorno
- Primitiva
- Herramienta de encaje
- Simulación funcional
- Simulación de temporización

INTRODUCCIÓN

En este capítulo, se analiza la arquitectura (organización y estructura interna) de los dispositivos SPLD, CPLD y FPGA. Se presentan varios CPLD específicos, incluyendo las familias MAX 7000 y MAX II de Altera y la familia CooleRunner II de Xilinx. Los dispositivos FPGA presentados son las familia Stratix de Altera y la familia Virtex de Xilinx.

Las explicaciones sobre las herramientas de desarrollo software cubren el flujo de diseño genérico para programar un dispositivo, incluyendo la introducción del diseño, la simulación funcional, la síntesis, la implementación, la simulación de temporización y la descarga. Asimismo, hay una sección dedicada al diagnóstico dentro de una placa de circuito una vez que el dispositivo programable está ya en operación. Los métodos de prueba incluyen la cama de pinchos, la sonda volante y la exploración de contorno.

■ ■ ■ APLICACIÓN A LOS SISTEMAS DIGITALES

La aplicación a los sistemas digitales ilustra un flujo genérico de diseño para programar la lógica necesaria para excitar un display de 7 segmentos. En la aplicación a los sistemas digitales del Capítulo 4 se ha desarrollado la lógica de cada uno de los segmentos y se han escrito los correspondientes programas VHDL para cada uno de ellos. Pueden introducirse los programas VHDL utilizando una herramienta software de introducción de texto. Sin embargo, ilustraremos el flujo de diseño basándonos en una técnica genérica de introducción de esquemáticos.

11.1 LÓGICA PROGRAMABLE: DISPOSITIVOS SPLD Y CPLD

Dos de los principales tipos simples de dispositivos de lógica programable (SPLD) son la PAL y la GAL. *PAL* significa dispositivo lógico de matriz programable (*Programmable Array Logic*) mientras que *GAL* significa dispositivo lógico de matriz genérica (*Generic Array Logic*). Generalmente, una PAL es un dispositivo programable una sola vez (OTP, *One-Time Programmable*) mientras que una GAL es un tipo de PAL, que es reprogramable; sin embargo, puesto que algunos dispositivos SPLD reprogramables se siguen denominando dispositivos PAL, la línea divisoria entre los dispositivos PAL y GAL es muy poco nítida dentro del uso común. El término *GAL* es una designación originalmente utilizada por Lattice Semicon-ductor y posteriormente licenciada a otros fabricantes. La estructura básica de los dispositivos PAL y GAL es una matriz AND programable junto con una matriz OR fija, lo cual constituye una arquitectura básica de tipo suma de productos. Los dispositivos complejos de lógica programable (CPLD, *Complex Programmable Logic Device*) son, básicamente, un único dispositivo con múltiples SPLD que proporciona más capacidad, para diseños lógicos de mayor envergadura.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Describir el funcionamiento de los SPLD. ■ Mostrar cómo se implementa una expresión suma de productos en una PAL o en una GAL. ■ Explicar los diagramas lógicos simplificados PAL/GAL. ■ Describir una macrocelda básica PAL/GAL. ■ Explicar la PAL16V8 y la GAL22V10. ■ Describir un dispositivos CPLD básico.

SPLD: la PAL

Una ***PAL*** (*Programmable Array Logic*, dispositivo lógico de matriz programable) consta de una matriz programable de puertas AND que se conecta a una matriz fija de puertas OR. Generalmente, las PAL se implementan con tecnología de proceso basada en fusibles y es, por tanto, programable una sola vez (OTP).

La estructura PAL permite implementar cualquier expresión lógica de tipo suma de productos (SOP, *sum-of-products*) con un número definido de variables. Como hemos visto anteriormente, cualquier función de lógica combinacional puede expresarse en forma SOP. En la Figura 11.1 se muestra una estrcutura PAL simple para dos variables de entrada y una salida, la mayoría de las PAL tienen múltiples entradas y múltiples salidas. Como hemos visto en el Capítulo 3, una matriz programable es, esencialmente, una cuadrícula o matriz de conductores que forma filas y columnas, existiendo un enlace programable en cada punto de cruce.

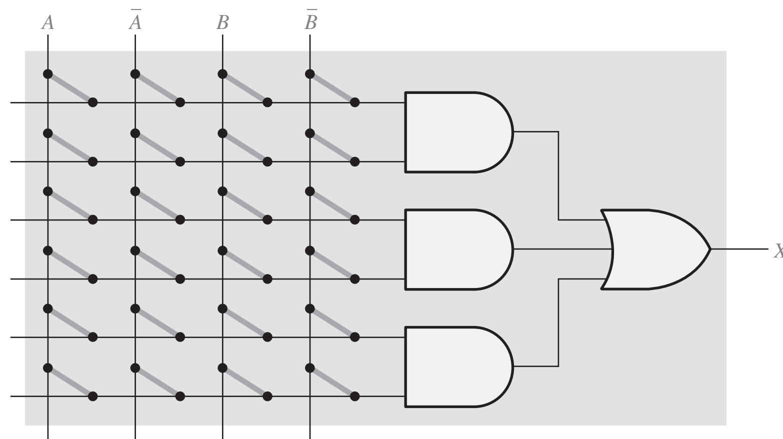


FIGURA 11.1 Estructura básica AND/OR de una PAL.

Cada conexión programable, que es un fusible en el caso de una PAL, se denomina *celda.* Cada fila está conectada a la entrada de una puerta AND y cada columna está conectada a una variable de entrada o a su complemento. Programando la presencia o ausencia de una conexión de tipo fusible, puede aplicarse cualquier combinación de variables de entrada o de complementos de las mismas a una puerta AND para formar cualquier término producto deseado. Las puertas AND están conectadas a una puerta OR, creando una salida de tipo suma de productos.

Implementación de una expresión suma de productos. En la Figura 11.2 se muestra cómo se programaría un ejemplo de PAL simple, de modo que el término producto AB es generado por la puerta AND superior, el término $A\overline{B}$ se genera en la puerta AND intermedio y el término $\overline{A}\overline{B}$ se genera en la puerta AND inferior. Como puede verse, los fusibles se dejan intactos para conectar las variables deseadas o sus complementos a las entradas apropiadas de la puerta AND. Los fusibles se queman cuando no se quiere utilizar una variable o su complemento en un determinado término producto. La salida final de la puerta OR es la expresión suma de productos.

$X = AB + A\overline{B} + \overline{A}\overline{B}$

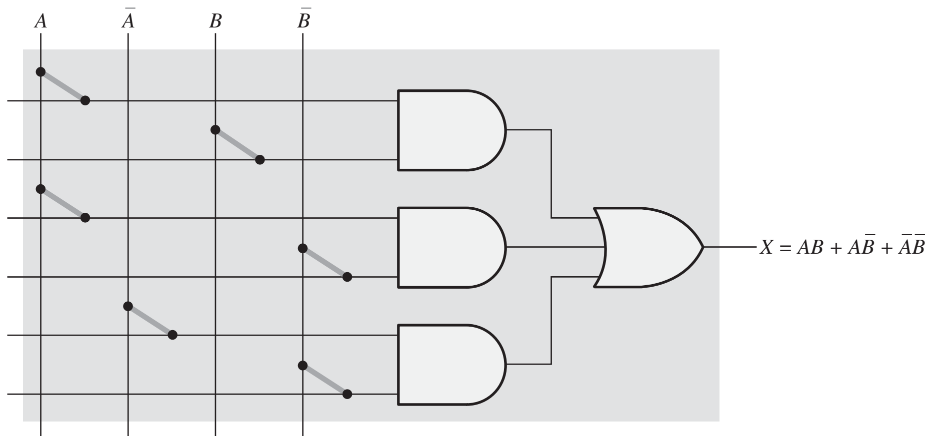


FIGURA 11.2 Implementación mediante PAL de una expresión suma de productos.

SPLD: la GAL

La ***GAL*** es esencialmente una PAL que puede reprogramarse. Tiene el mismo tipo de organización AND/OR que la PAL. La diferencia básica es que la GAL utiliza una tecnología de proceso reprogramable, como por ejemplo EEPROM (E^2CMOS), en lugar de emplear fusibles, como se muestra en la Figura 11.3.

Notación simplificada para diagramas PAL/GAL

Los dispositivos PAL y GAL reales tienen numerosas puertas AND y OR además de otros elementos, y son capaces de aceptar muchas variables, junto con sus complementos. La mayoría de los diagramas PAL y GAL que podremos ver en una hoja de características utilizan notación simplificada, como la ilustrada en la Figura 11.4, para evitar que los esquemáticos se compliquen demasiado.

Las variables de entrada a una PAL o a una GAL suelen pasarse por un *buffer* para evitar cargarlas con el gran número de entradas de puertas AND a las que están conectadas. En el diagrama, el símbolo de triángulo representa un *buffer* que genera tanto la variable como su complemento. Las conexiones fijas de las variables de entrada y *buffer* se muestran utilizando la notación estándar de punto.

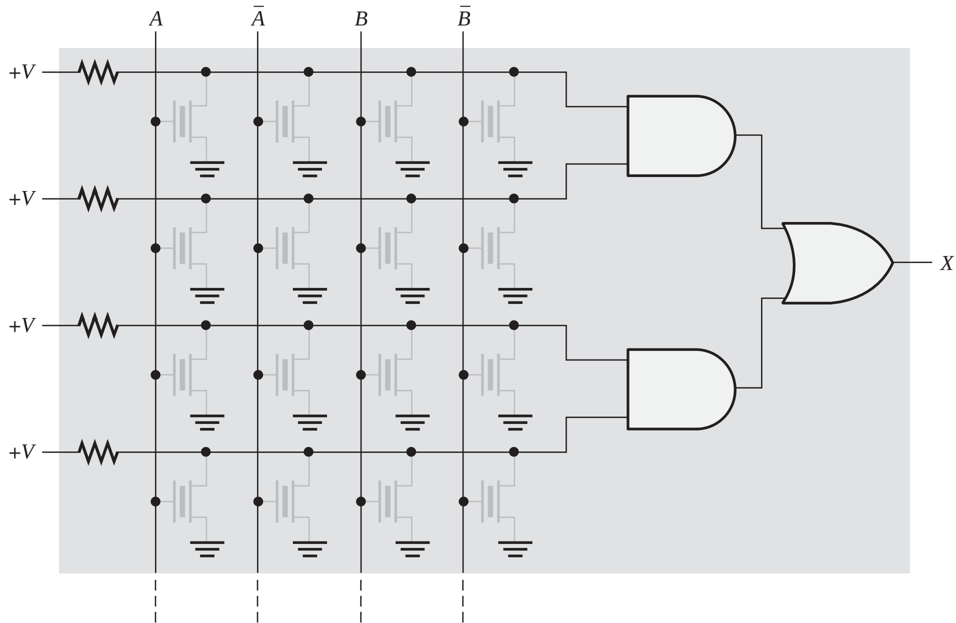


FIGURA 11.3 Matriz GAL simplificada.

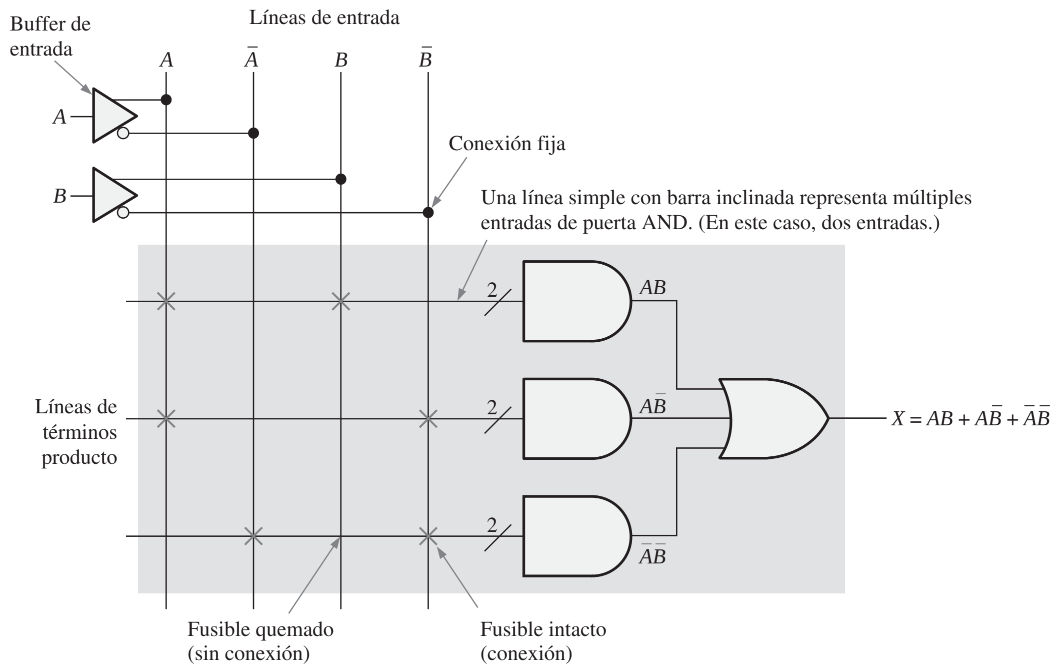


FIGURA 11.4 Una parte de una PAL/GAL programada.

Los dispositivos PAL y GAL tienen un gran número de líneas de interconexión programables, y cada puerta AND tiene múltiples entradas. Los diagramas lógicos típicos de los dispositivos PAL y GAL representan

las puertas AND de múltiples entradas mediante un símbolo de puerta AND que tiene una única línea de entrada con una barra inclinada y un dígito que indica el número real de entradas. La Figura 11.4 ilustra esta notación para el caso de puertas AND de 2 entradas.

Los enlaces programables de una matriz se indican en el diagrama mediante una X mayúscula en el punto de cruce para los fusibles intactos (o para otros tipos de enlaces) y la ausencia de una X indica un fusible quemado o la ausencia de otro tipo de conexión. En la Figura 11.4 está programada la siguiente función lógica de dos variables:

$$X = AB + A\overline{B} + \overline{A}\overline{B}.$$

EJEMPLO 11.1

Mostrar cómo se programaría una PAL para la siguiente función lógica de 3 variables:

$$X = A\overline{B}C + \overline{A}B\overline{C} + \overline{A}\overline{B} + AC$$

Solución En la Figura 11.5 se muestra la matriz programada. Los enlaces fusibles intactos se indican mediante X. La ausencia de X indica que el fusible está abierto.

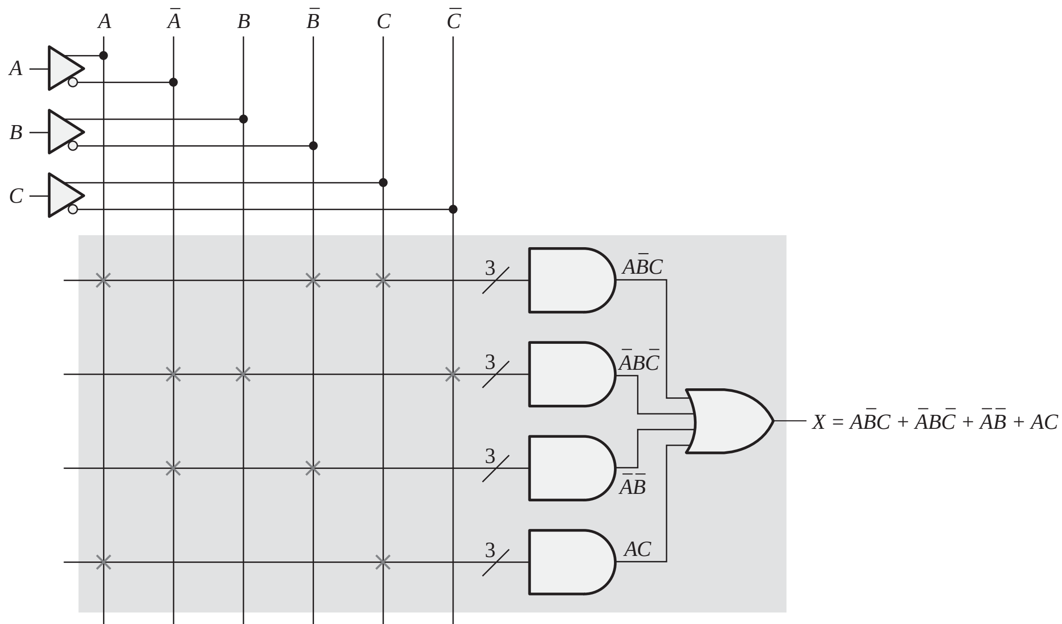


FIGURA 11.5

Problema relacionado* Escribir la expresión de salida si se abren también los enlaces fusibles que conectan la entrada *A* a la fila superior y a la fila inferior en la Figura 11.5.

* Las respuestas se encuentran al final del capítulo.

Diagrama de bloques general de una PAL/GAL

En la Figura 11.6 se muestra un diagrama de bloques de una PAL o GAL. Recuerde que la diferencia básica es que una GAL tiene una matriz reprogramable mientras que la PAL sólo puede programarse una vez. Las salidas de la matriz AND programable están conectadas a puerta OR fijas que a su vez están conectadas a circuitos lógicos adicionales de salida. El conjunto formado por una puerta OR y su lógica de salida asociada suele denominarse *macrocelda.* La complejidad de la macrocelda depende de cada dispositivo concreto, y en los dispositivos GAL las macroceldas suelen ser reprogramables.

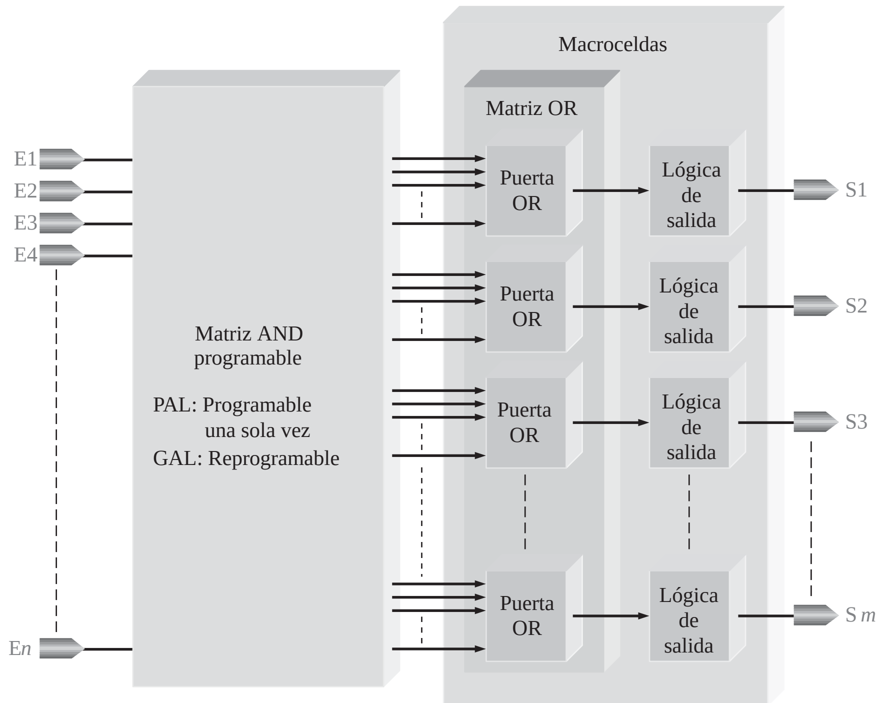


FIGURA 11.6 Diagrama de bloques general de una PAL o GAL.

Macroceldas

Generalmente, una ***macrocelda*** consiste en una puerta OR y cierta lógica de salida asociada. Las macroceldas varían en su complejidad dependiendo del tipo concreto de PAL o GAL. Una macrocelda puede configurarse para implementar lógica combinacional, lógica registrada o una combinación de ambos tipos de lógica. La lógica ***registrada*** significa que existe un flip-flop en la celda que permite implementar funciones de lógica secuencial. La operación registrada de las macroceldas se trata en la Sección 11.4.

La Figura 11.7 ilustra tres tipos básicos de macroceldas con lógica combinacional. La parte (a) muestra una macrocelda simple con la puerta OR y un inversor con control triestado, que puede hacer que el inversor actúe como un circuito abierto para desconectar completamente la salida. La salida del inversor triestado puede estar a nivel BAJO, a nivel ALTO o desconectada. La parte (b) es una macrocelda que puede actuar como entrada o como salida. Cuando la salida se emplea como entrada, se desconecta el inversor triestado y la entrada va al *buffer* que está conectado a la matriz AND. La parte (c) es una macrocelda que puede programarse para tener una salida activa a nivel ALTO o activa a nivel BAJO, o que puede utilizarse como entrada. Una entrada a una puerta OR-exclusiva (XOR) puede programarse para estar a nivel ALTO o BAJO. Cuando

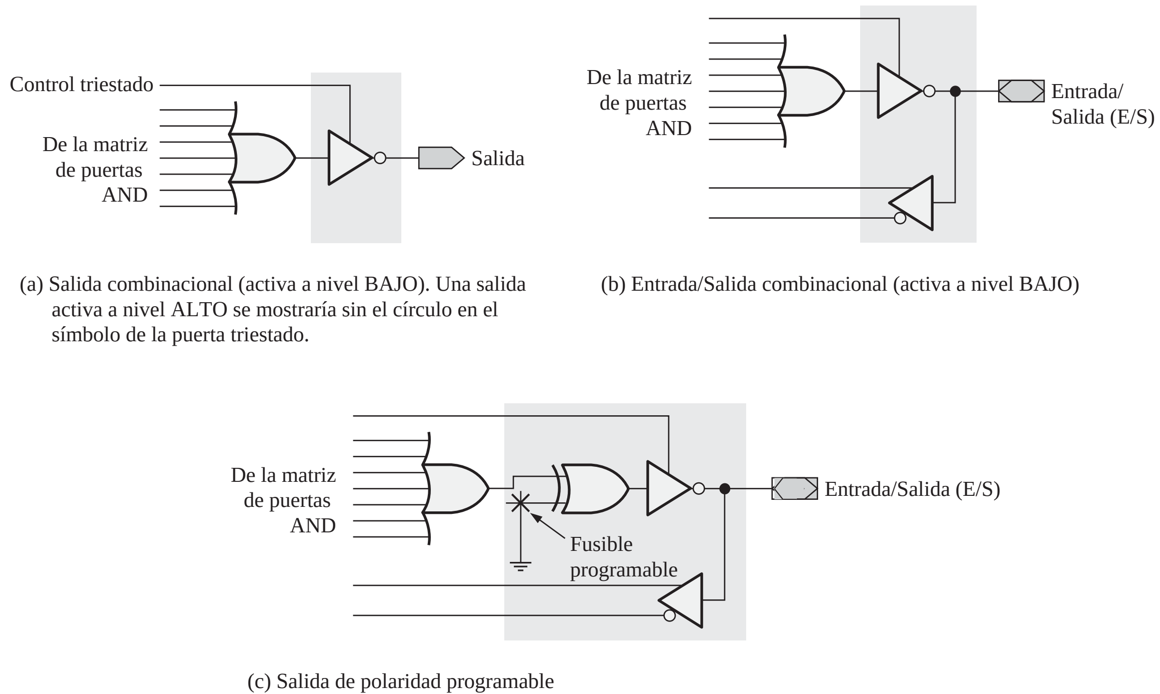


FIGURA 11.7 Tipos básicos de macroceldas PAL/GAL para lógica combinacional.

la entrada programable de la puerta XOR está a nivel ALTO, la salida de la puerta OR se invierte porque $0 \oplus 1 = 1$ y $1 \oplus 1 = 0$. De forma similar, cuando la entrada programable de la puerta XOR está a nivel BAJO, la salida de la puerta OR no se invierte porque $0 \oplus 0 = 0$ y $1 \oplus 0 = 1$.

Dispositivos SPLD

Generalmente, los encapsulados de los SPLD suelen tener entre 20 y 28 pines. Dos factores que pueden usarse para determinar si una cierta PAL o GAL resulta adecuada para un diseño lógico determinado son el número de entradas y salidas y el número de puertas equivalentes o densidad del dispositivo. Otros parámetros que hay que considerar son la frecuencia máxima de operación, los retardos y la tensión continua de alimentación. Entre las diversas empresas que fabrican dispositivos SPLD podemos citar Lattice, Actel, Atmel y Cypress. Diversos fabricantes de SPLD pueden utilizar distintas formas de definir la densidad de los dispositivos, por lo que hay que tener en cuenta esta variabilidad a la hora de utilizar el número especificado de puertas equivalentes.

Dos tipos comunes de dispositivos PAL y GAL son el 16V8 y el 22V10. La designación de los dispositivos indica el número de entradas, el número de salidas y el tipo de lógica de salida. Por ejemplo, 16V8 significa que el dispositivo tiene dieciséis entradas, ocho salidas y que estas salida son variables (V). La letra *L* o *H* significa que la salida es activa a nivel bajo (*L*, *Low*, bajo) o a nivel alto (*H*, *High*, alto). En la Figura 11.8 se muestra el diagrama de bloques de una PAL16V8 y un encapsulado de SPLD típico.

Cada macrocelda tiene ocho entradas procedentes de la matriz de puertas AND, por lo que puede haber hasta ocho términos producto para cada salida. Existen diez entradas dedicadas (E), dos salidas dedicadas (S) y seis pines que pueden utilizarse como entradas o salidas (E/S). Cada salida es activa a nivel BAJO. La PAL16V8 tiene una densidad de aproximadamente 300 puertas equivalentes.

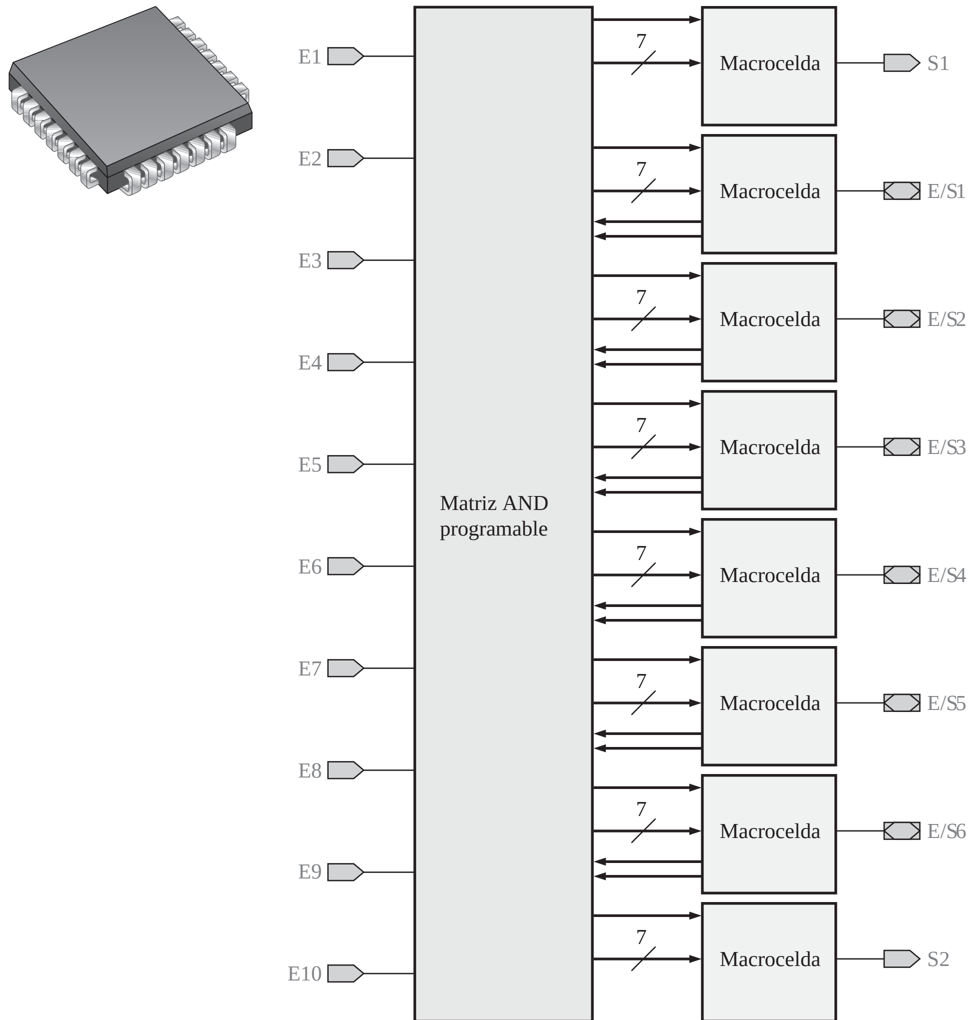


FIGURA 11.8 Diagrama de bloques lógicos de una PAL16V8 y encapsulado de un SPLD típico.

En la Figura 11.9 se muestra el diagrama de bloques de una GAL22V10 y un encapsulado SPLD típico. Este dispositivo tiene doce entradas dedicadas y diez pines que pueden actuar como entradas o como salidas. Las macroceldas tienen entradas procedentes de la matriz AND que varían entre ocho y dieciséis como se indica mediante la notación simplificada. La GAL22V10 tiene una densidad de aproximadamente 500 puertas equivalentes.

Dispositivos CPLD

Un dispositivo ***CPLD*** (*Complex Programmable Logic Device*) consta básicamente de múltiples matrices SPLD con interconexiones programables, como se ilustra en la Figura 11.10. Aunque la organización interna de los dispositivos CPLD varía de unos fabricantes a otros, la Figura 11.10 representa un dispositivo CPLD genérico. Cada una de las matrices SPLD de un CPLD se denomina ***LAB*** (*Logic Array Block*, bloque de matriz lógica). También se utilizan otras designaciones, *bloque funcional*, *bloque lógico* o *bloque genérico.*

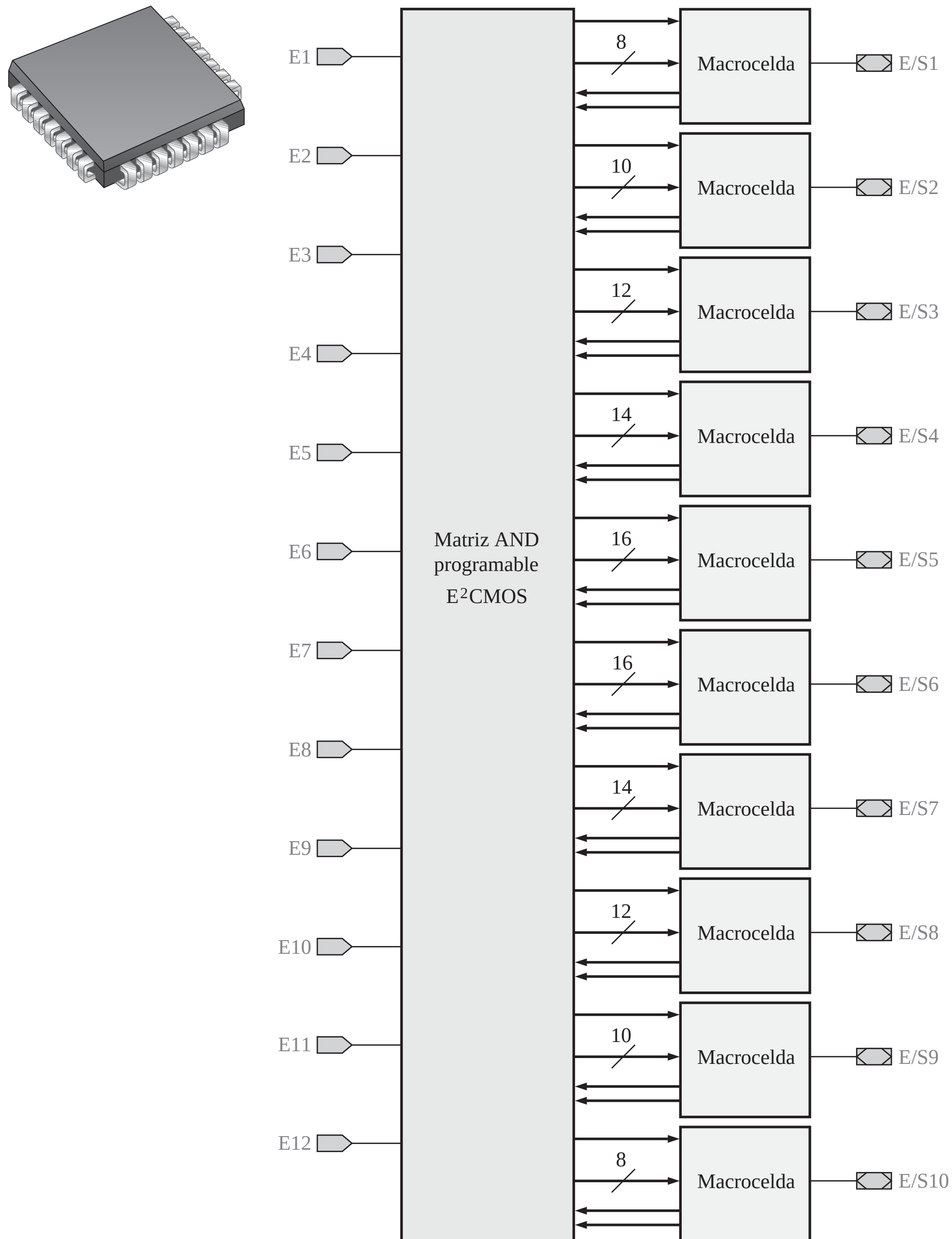


FIGURA 11.9 Diagrama de bloques de la GAL 22V10 y encapsulado SPLD típico.

Las interconexiones programables se denominan normalmente *PIA* (*Programmable Interconnect Array*, matriz de interconexiones programables), aunque algunos fabricantes, como Xilinx, utilizan el término *AIM* (*Advanced Interconnect Matrix*, matriz avanzada de interconexión) o alguna otra designación similar. Los bloques LAB y las interconexiones entre esos bloques se programan mediante software. Un CPLD puede programarse para implementar funciones lógicas complejas basadas en la estructura suma de productos de los blo-

ques LAB individuales (que en realidad son dispositivos SPLD). Las entradas pueden conectarse a cualquiera de los bloques LAB y sus salidas pueden interconectarse a cualquier otro bloque LAB a través de la PIA.

La mayoría de los fabricantes de lógica programable suministran una serie de dispositivos CPLD que varían en cuanto a su densidad, tecnología de proceso, consumo de potencia, tensión de alimentación y velocidad. Los fabricantes suelen especificar la densidad de un CPLD en términos del número de macroceldas o de bloques de matriz lógica. Dichas densidades pueden ir desde las decenas de macroceldas hasta más de 2000 macroceldas con encapsulados de varios cientos de pines. A medida que los dispositivos PLD vayan haciéndose más complejos, las densidades máximas también se incrementarán. La mayoría de los dispositivos CPLD son reprogramables y emplean tecnología de proceso EEPROM o SRAM para implementar los enlaces programables. El consumo de potencia puede ir desde unos pocos milivatios hasta unos pocos cientos de milivatios, mientras que las tensiones continuas de alimentación están comprendidas entre 2,5 V y 5 V, dependiendo de cada dispositivo específico.

Son varios los fabricantes, como por ejemplo, Altera, Xilinx, Lattice y Cypress, que suministran dispositivos CPLD. En este capítulo vamos a centrarnos en los productos de Altera y de Xilinx, porque son dos de las principales empresas en este sector. Otras empresas ofrecen dispositivos y software similares, por lo que puede fácilmente realizarse la transición a otros productos una vez que uno se ha familiarizado con uno o dos de ellos. Como veremos, los CPLD y otros dispositivos de lógica programable son, en realidad, una combinación de hardware y software.

REVISIÓN DE LA SECCIÓN 11.1

Las respuestas se encuentran al final del capítulo

1. ¿Qué quiere decir PAL?
2. ¿Qué quiere decir GAL?
3. ¿Cuál es la diferencia entre una PAL y una GAL?
4. Básicamente, ¿qué es lo que contiene una macrocelda?
5. ¿Qué es un CPLD?

11.2 DISPOSITIVOS CPLD DE ALTERA

Altera fabrica varias familias de dispositivos CPLD incluyendo las familias MAX II, MAX 3000 y Max 7000. En esta sección, nos vamos a centrar principalmente en la familia MAX 7000 para ilustrar los conceptos de la arquitectura tradicional de los CPLD, teniendo siempre presente que otras familias pueden variar hasta cierto punto en lo que respecta a su arquitectura y/o en lo que respecta a parámetros tales como la densidad, la tecnología de proceso, el consumo de potencia, la tensión de alimentación y la velocidad.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Describir un CPLD típico de la familia MAX. ■ Explicar la arquitectura básica de los CPLD de las familias MAX 7000 y MAX II. ■ Explicar cómo se generan términos producto en los dispositivos CPLD.

CPLD MAX 7000

La **arquitectura** de un CPLD es la forma en que están organizados y dispuestos los elementos internos del dispositivo. La arquitectura de la familia MAX 7000 es similar al diagrama de bloques de un CPLD genérico (mostrado en la Figura 11.10). Tiene la estructura clásica PAL/GAL que permite generar funciones suma de producto. La densidad varía entre 2 bloques LAB y 16 bloques LAB, dependiendo del dispositivo concreto de la serie que se utilice. Recuerde que un bloque LAB es aproximadamente equivalente a un SPLD y que el

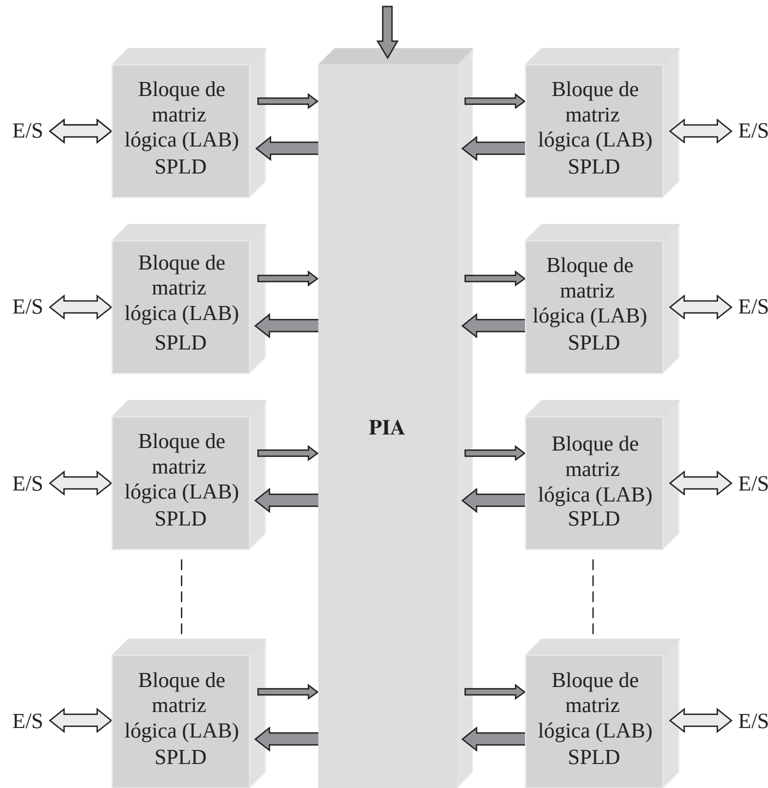


FIGURA 11.10 Diagrama de bloques básico de un CPLD genérico.

tamaño de los encapsulados varían entre 44 y 208 pines. La serie MAX 7000 de dispositivos CPLD utiliza una tecnología de proceso basada en EEPROM. Las versiones programables dentro del sistema (ISP, *In-System Programmable*) utilizan la interfaz estándar JTAG.

La Figura 11.11 muestra un diagrama de bloques general de los CPLD de la serie MAX 7000 de Altera. Se muestran cuatro bloques LAB, pero puede haber hasta 16, dependiendo del dispositivo concreto de la serie. Cada uno de los cuatro bloques LAB está compuesto por 16 macroceldas y los distintos bloques LAB se interconectan a través de la PIA, que es una estructura programable de bus global (se conecta a todos los bloques LAB) a la que se conectan las entradas de propósito general, las líneas de E/S y las macroceldas.

La macrocelda. En la Figura 11.12 se muestra un diagrama simplificado de una macrocelda de la serie MAX 7000. La macrocelda contiene una pequeña matriz AND programable con cinco puertas AND, una puerta OR, una matriz de selección de términos producto para conectar las salidas de las puertas AND a la puerta OR y una lógica asociada que puede programarse para actuar como entrada, como salida de lógica combinacional o como salida registrada. Esta macrocelda se estudia en más detalle en la Sección 11.4.

Aunque basada en los mismos conceptos, esta macrocelda difiere en cierta medida de la macrocelda presentada en la Sección 11.1 al hablar de los dispositivos CPLD, porque contiene una parte de la matriz AND programable y una matriz de selección de términos producto. Como se muestra en la Figura 11.12, hay cinco puertas AND que suministran términos producto procedentes de la PIA a la matriz de selección de términos producto. El término producto generado por la puerta AND inferior puede ser re-invertido y realimentado hacia la matriz programable, como mecanismo de expansión compartido que puede ser utilizado por otras macroceldas. Las entradas expandidas paralelas permiten apropiarse de términos producto no utilizados de

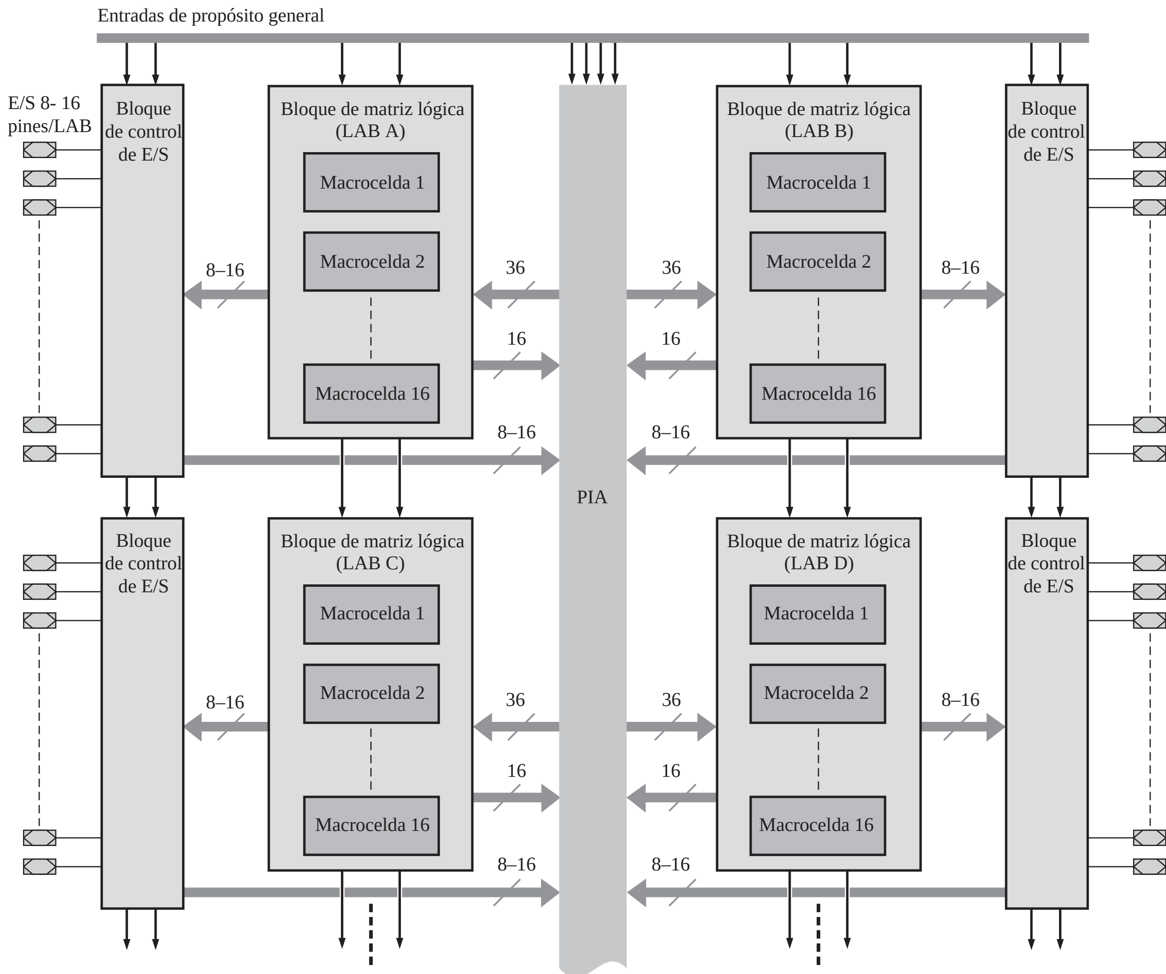


FIGURA 11.11 Diagrama de bloques básico de un CPLD de la serie MAX 7000 de Altera.

otras macroceldas, con el fin de expandir una expresión suma de productos. La matriz de selección de términos producto es una matriz de conexiones programables que se emplea para conectar a la puerta OR una serie de salidas seleccionadas de la matriz AND y de las entradas de expansión.

Líneas de expansión compartidas. En cada macrocelda de un bloque LAB hay disponible un término producto complementado que puede utilizarse para incrementar el número de términos producto en una expresión suma de productos. La Figura 11.13 ilustra la manera de utilizar un término de expansión compartido procedente de otra macrocelda con el fin de crear términos producto adicionales. En este caso, cada una de las cinco puertas AND de la matriz de una macrocelda está limitada a cuatro entradas y puede, por tanto, generar un término producto de hasta 4 variables, como se muestra en la parte (a). La Figura 11.13(b) muestra la expansión a dos términos producto.

Cada macrocelda MAX 7000 puede generar hasta cinco términos producto mediante su matriz AND. Si una macrocelda necesita más de cinco términos producto para generar su salida suma de productos, puede emplear un término de expansión procedente de otra macrocelda. Suponga que un determinado diseño requiere una suma de productos que requiere seis términos producto. La Figura 11.14 muestra cómo emplearse un

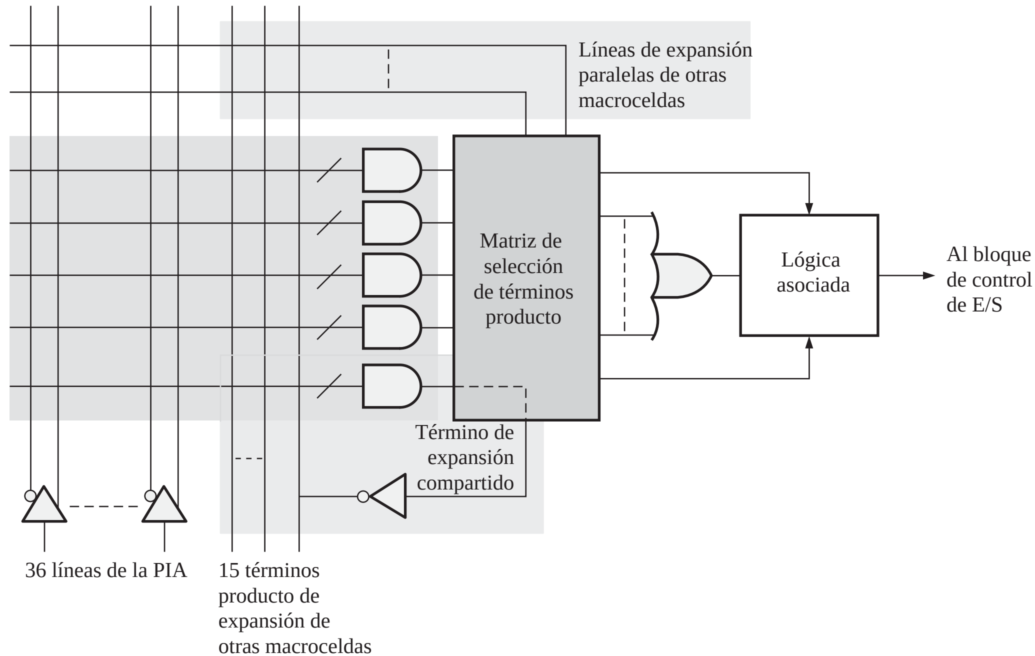


FIGURA 11.12 Diagrama simplificado de una macrocelda en un CPLD de la serie MAX 7000.

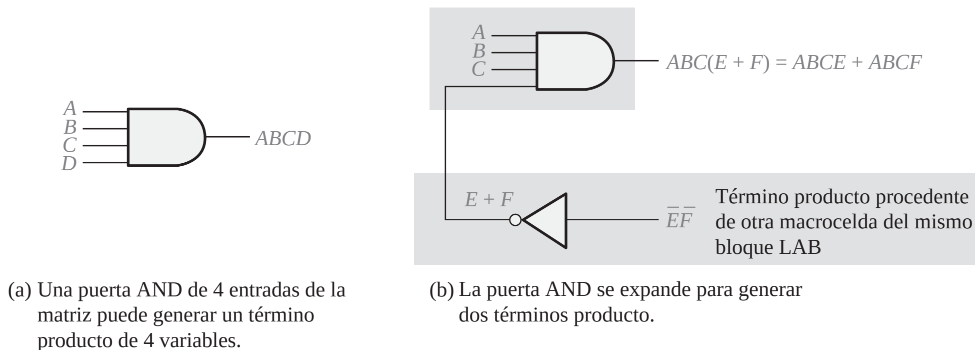


FIGURA 11.13 Ejemplo de utilización de un término de expansión compartido de una macrocelda para incrementar el número de términos producto.

término producto de otra macrocelda para incrementar una salida suma de productos. La macrocelda 2, que está infrautilizada, genera un término de expansión compartido ($E + F$) que se conecta a la quinta puerta AND de la macrocelda 1, con el fin de generar una expresión suma de producto con seis términos. Las X y las líneas representan las conexiones implementadas en el hardware por el compilador software que ejecuta el diseño programado.

Términos de expansión paralelos. Otra forma de incrementar el número de términos producto de una macrocelda consiste en utilizar términos de expansión paralelos, en los que los términos producto adicionales se combinan mediante una operación OR con los términos generados en la macrocelda en lugar de combinarse en la matriz AND, como sucede en los términos de expansión compartidos. Cada macrocelda puede apropiarse de términos producto no utilizados en las macroceldas vecinas (hasta cinco términos producto de otras tres

Macrocelda 1

$ABCD + \bar{A}BCD + \bar{A}BC\bar{D}$
$+ A\bar{B}CD + ABCE + ABCF$

Términos de expansión

Matriz de selección de términos producto

Macrocelda 2

$\bar{A}\bar{B}CD + ABCD + \bar{A}B\bar{C}D$

$\bar{E}\bar{F}$

Matriz de selección de términos producto

Término de expansión $E + F$ hacia macrocelda 1

A B C D E F

FIGURA 11.14 Ilustración simplificada que muestra la utilización de un término de expansión compartido de otra macrocelda con el fin de completar una expresión suma de productos.

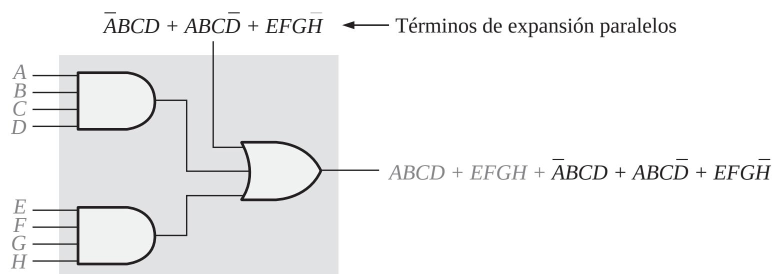


FIGURA 11.15 Concepto básico de expansión en paralelo.

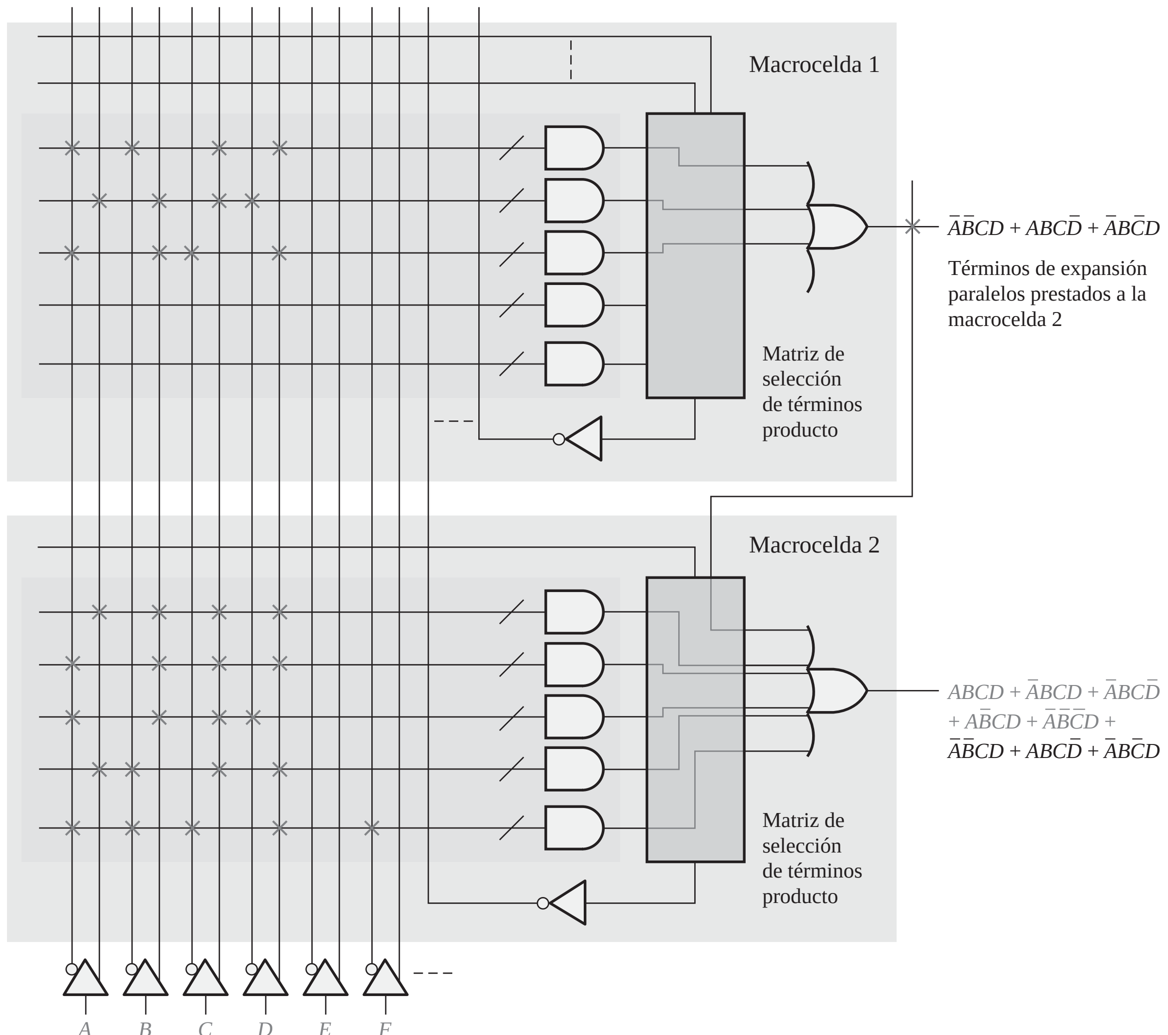


FIGURA 11.16 Ilustración simplificada del uso de términos de expansión paralelos procedentes de otra macrocelda con el fin de incrementar una expresión suma de productos.

macroceldas en los dispositivos MAX 7000). El concepto básico se ilustra en la Figura 11.15, en donde un circuito simplificado que puede generar dos términos producto se apropia de tres términos producto adicionales.

La Figura 11.16 muestra la forma en que una macrocelda puede apropiarse de términos de expansión paralelos procedentes de otra macrocelda, con el fin de expandir la salida suma de productos. La macrocelda 2 utiliza tres términos producto de la macrocelda 1 para generar una expresión suma de productos de ocho términos. Las X y las líneas representan las conexiones implementadas en el hardware por el compilador software que ejecuta el diseño programado.

El CPLD MAX II

La arquitectura del CPLD MAX II difiere enormemente de la familia MAX 7000 y es lo que Altera denomina un dispositivo CPLD "post-macrocelda". Como se muestra en el diagrama de bloques de la Figura 11.17,

este dispositivo contiene bloques de matriz lógica (LAB) cada uno de los cuales está compuesto por múltiples elementos lógicos (LE, *Logic Element*). El LE es la unidad básica de diseño lógico y su papel es análogo al de la macrocelda. Las interconexiones programables están organizadas según una disposición de filas y columnas que se extienden entre los bloques LAB, y los elementos de E/S (IOE, *Input/Output Element*) están orientados alrededor del perímetro. La arquitectura de esta familia de dispositivos CPLD es similar a la de los dispositivos FPGA de los que hablaremos en la Sección 11.5. De hecho, podemos considerar un dispositivo MAX II como una FPGA de baja densidad.

Una diferencia fundamental entre el CPLD MAX II y el CPLD clásico basado en estructura SPLD es la forma en que se desarrollan las funciones lógicas. Los dispositivos MAX II utilizan tablas de consulta (LUT, *Look-Up Table*) en lugar de matrices AND/OR. Una ***LUT*** es, básicamente, un tipo de memoria que puede programarse para generar sumas de productos (lo que se verá más en detalle en la Sección 11.5). Estos dos distintos enfoques se comparan en la Figura 11.18.

Como ya hemos mencionado, un CPLD MAX II tiene una disposición de las interconexiones en forma de filas/columnas, en lugar de las interconexiones de tipo canal que podemos encontrar en la mayoría de los CPLD clásicos. Estos dos distintos enfoques se comparan en la Figura 11.19 y pueden entenderse comparando las Figuras 11.11 y 11.17.

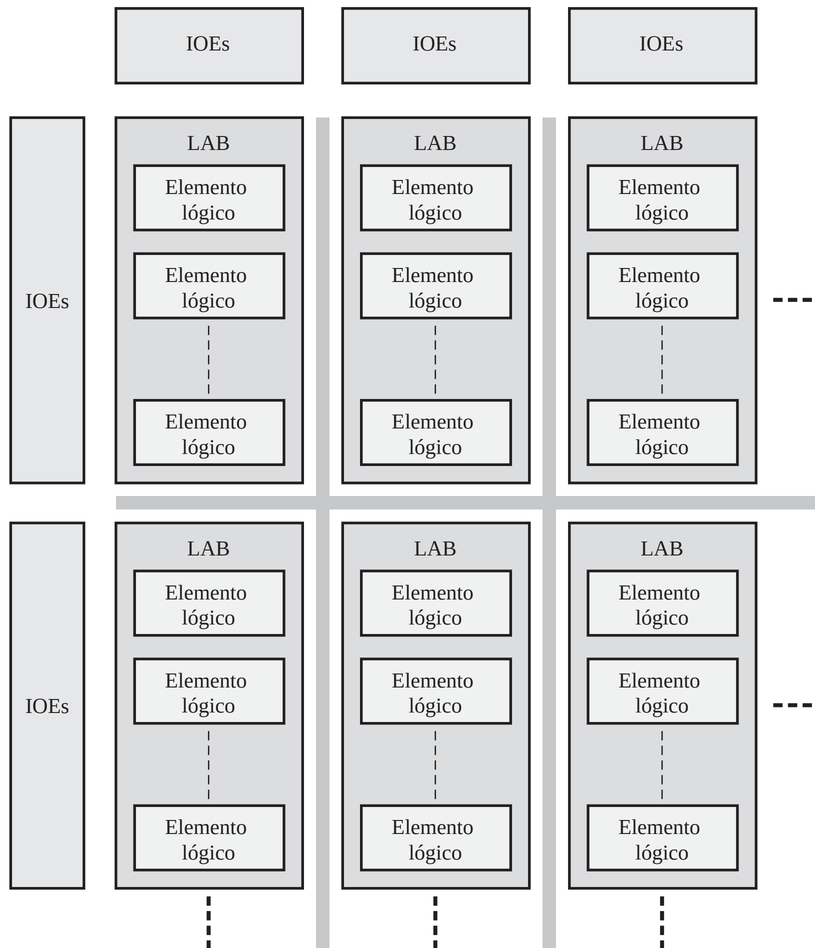


FIGURA 11.17 Diagrama de bloques simplificado del CPLD MAX II.

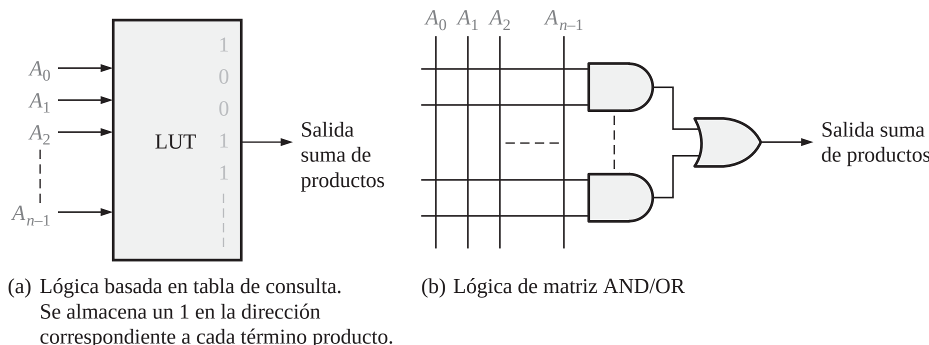


FIGURA 11.18 Los dispositivos CPLD MAX II tienen lógica basada en bloques LUT. Los dispositivos CPLD clásicos tienen matrices AND/OR.

La mayoría de los CPLD utiliza una tecnología de proceso no volátil para implementar las conexiones programables. Sin embargo, MAX II emplea una tecnología de proceso basada en SRAM que es **volátil**, por lo que toda la lógica programada se pierde en el momento de desconectar la alimentación. La memoria integrada en el chip almacena los datos del programación utilizando tecnología de memoria no volátil y se encarga de reconfigurar el CPLD en el momento en que se conecta de nuevo la alimentación.

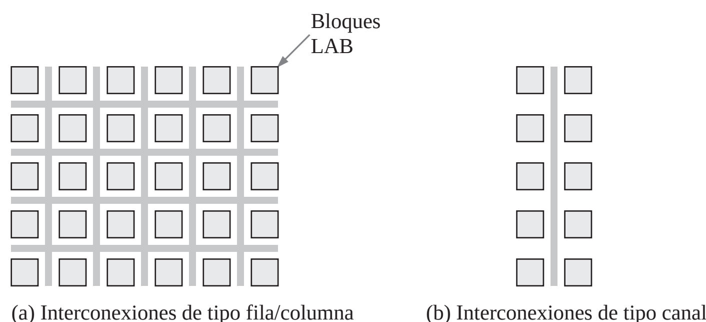


FIGURA 11.19 Los dispositivos CPLD MAX II tienen interconexiones de tipo fila/columna. Los dispositivos CPLD clásicos tienen interconexiones de tipo canal.

REVISIÓN DE LA SECCIÓN 11.2

1. ¿Qué significa LAB?
2. Describa un bloque LAB de los dispositivos CPLD MAX 7000.
3. ¿Para qué sirve un término de expansión compartido?
4. ¿Para qué sirve un término de expansión paralelo?
5. ¿Cuáles son las diferencias entre un dispositivo MAX II y un dispositivo MAX 7000?

11.3 DISPOSITIVOS CPLD DE XILINX

Xilinx, al igual que Altera, fabrica una serie de dispositivos CPLD que varían en cuanto a densidad, tecnología de proceso, consumo de potencia, tensión de alimentación y velocidad. Xilinx fabrica diver-

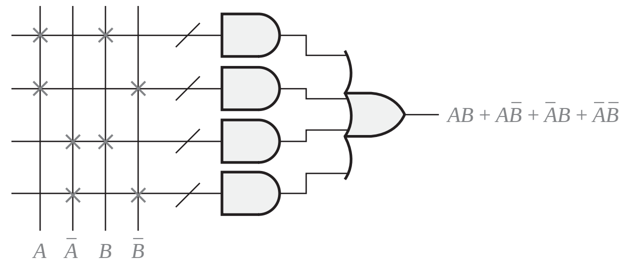
sas familias de CPLD, incluyendo CoolRunner II, CoolRunner XPLA3 y XC9500. La familia XC9500 es similar en cuando a su arquitectura a la familia de dispositivos CPLD MAX 7000 de Altera y presenta la arquitectura clásica PAL/GAL. En esta sección vamos a centrarnos exclusivamente en la familia CoolRunner II con el fin de ilustrar los conceptos, teniendo siempre presente que las otras series pueden variar en cierta medida en cuanto a su arquitectura y/o en cuanto a los parámetros anteriormente mencionados. Esta familia de dispositivos CPLD es programable dentro del sistema y compatible con el estándar JTAG.

Al finalizar esta sección, el lector deberá ser capaz de:

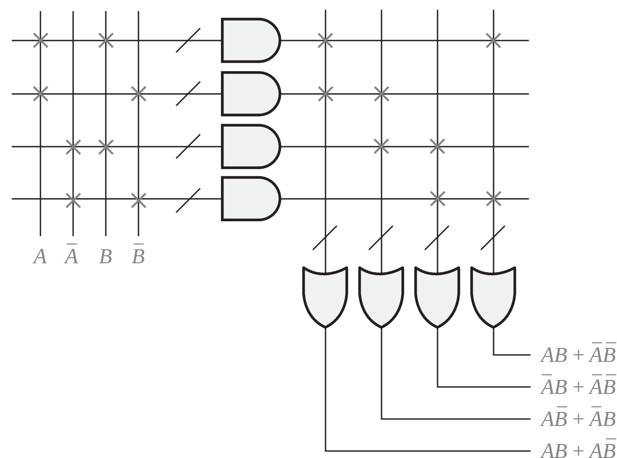
■ Describir una PLA y compararla con una PAL. ■ Explicar la arquitectura de los CPLD CoolRunner II. ■ Describir un bloque funcional.

PLA (*Programmable Logic Array*)

Como ya hemos visto, la arquitectura de un CPLD es la forma en que se organizan y disponen los elementos internos del dispositivo. La arquitectura de la familia CoolRunner II de Xilinx está basada en una PLA (*Programmable Logic Array*, matriz de lógica programable) en lugar de en una estructura PAL (*Program mable Array Logic*, dispositivo lógico de matriz programable). La Figura 11.20 compara una estructura PAL simple con una estructura PLA simple. Como ya sabemos, la PAL tiene una matriz AND programable



(a) Matriz de tipo PAL



(b) Matriz de tipo PLA

FIGURA 11.20 Comparación de una PLA básica y una PAL básica.

seguida de una matriz OR fija y genera una expresión suma de productos, como se muestra en el ejemplo de la Figura 11.20(a). La matriz **PLA** tiene una matriz AND programable seguida de una matriz OR programable, como se muestra en el ejemplo de la Figura 11.20(b).

CoolRunner II

El dispositivo CPLD CoolRunner II utiliza una estructura de tipo PLA. Este dispositivo tiene múltiples bloques funcionales (FB, *Function Block*), que son análogos a los bloques LAB de la familia MAX 7000 de Altera (Figura 11.11). Cada bloque funcional contiene dieciséis macroceldas al igual que el bloque LAB. Los bloques funcionales se interconectan mediante una matriz avanzada de interconexión (AIM), que es análoga

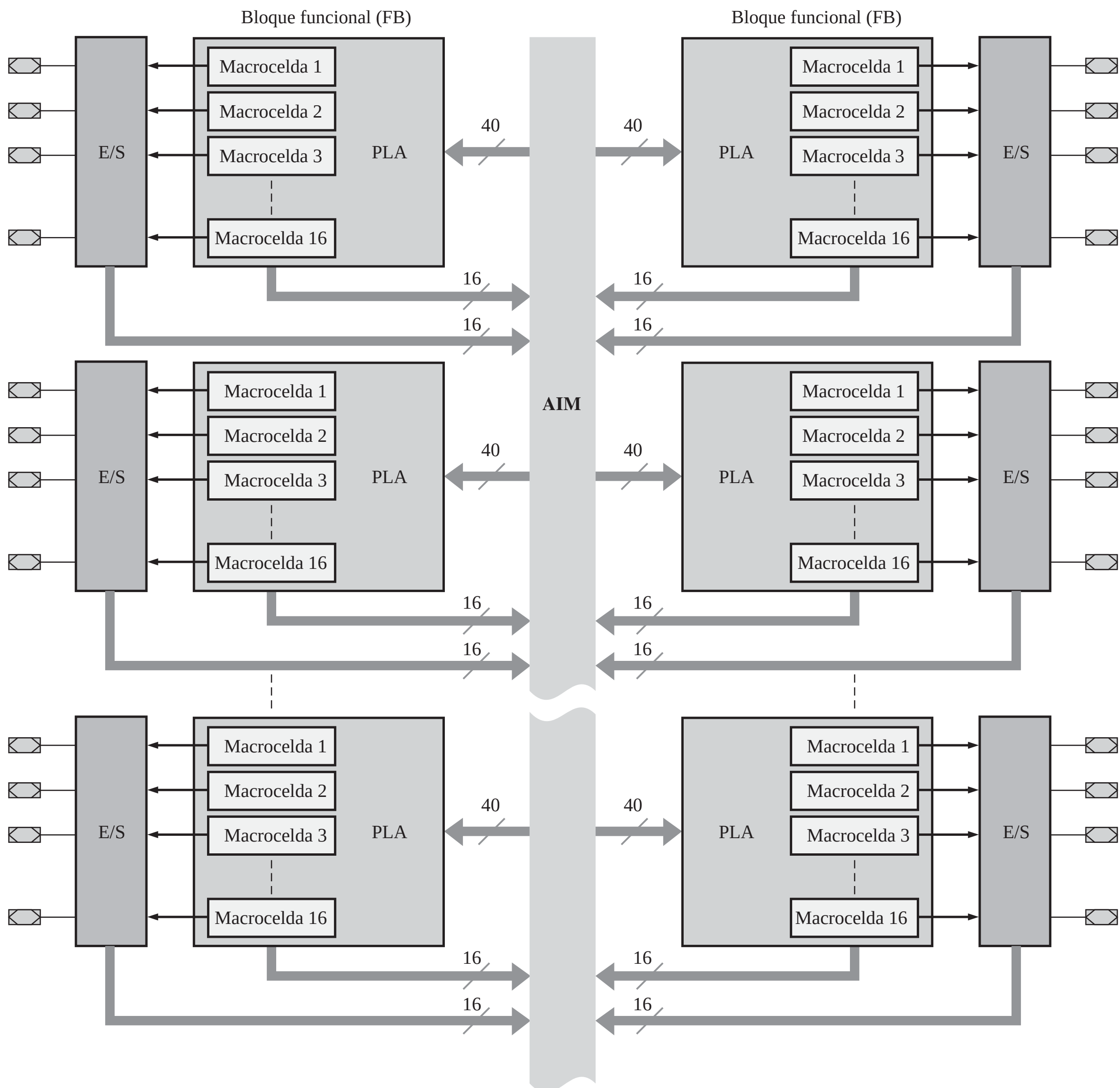


FIGURA 11.21 Diagrama de bloques básico de la arquitectura de los dispositivos CPLD CoolRunner II de Xilinx.

a la matriz PIA de la serie MAX 7000. En la Figura 11.21 se muestra un diagrama de bloques básico de la arquitectura de los dispositivos CoolRunner II. Como puede ver, desde el punto de vista de un diagrama de bloques básico, no existe mucha diferencia entre los CPLD de Xilinx y de Altera; sin embargo, sí que existen diferencias internas.

La serie CoolRunner II de dispositivos CPLD contiene entre 32 y 512 macroceldas. Puesto que existen dieciseis macroceldas por cada bloque funcional, el número de bloques funcionales va desde 2 a 32. En la Figura 11.22 se muestra un diagrama enormemente simplificado de un bloque funcional (FB). La matriz AND programable tiene 56 puertas AND y la matriz OR programable tiene 16 puertas OR. Con la estructura de tipo PLA, cualquier término producto puede conectarse a cualquier puerta OR para crear una salida suma de productos. Con una utilización máxima, cada FB puede generar 15 salidas suma de productos, cada una de las cuales puede tener 56 términos producto. Esta macrocelda se analiza en detalle en la Sección 11.4.

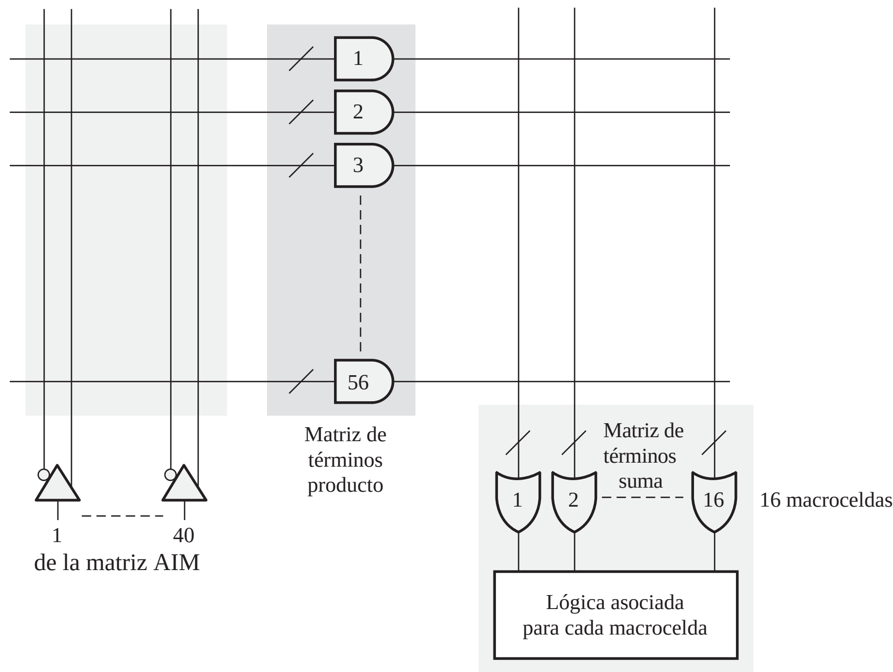


FIGURA 11.22 Diagrama simplificado de un bloque funcional (FB) de CoolRunner II con estructura PLA.

EJEMPLO 11.2

Muestre las conexiones programadas en el FB simplificado de la Figura 11.22 necesarias para generar la siguiente función suma de productos a partir de la macrocelda 1: $ABCD + A\overline{B}C\overline{D} + AB\overline{C}D$ y la siguiente función suma de productos a partir de la macrocelda 2: $\overline{A}BC\overline{D} + \overline{A}BCD + A\overline{B}CD + ABC\overline{D}$.

Solución Las X en la Figura 11.23 indican conexiones programadas en las matrices AND y OR.

Problema relacionado ¿Cuántas funciones suma de productos pueden generarse mediante el FB de la Figura 11.23?

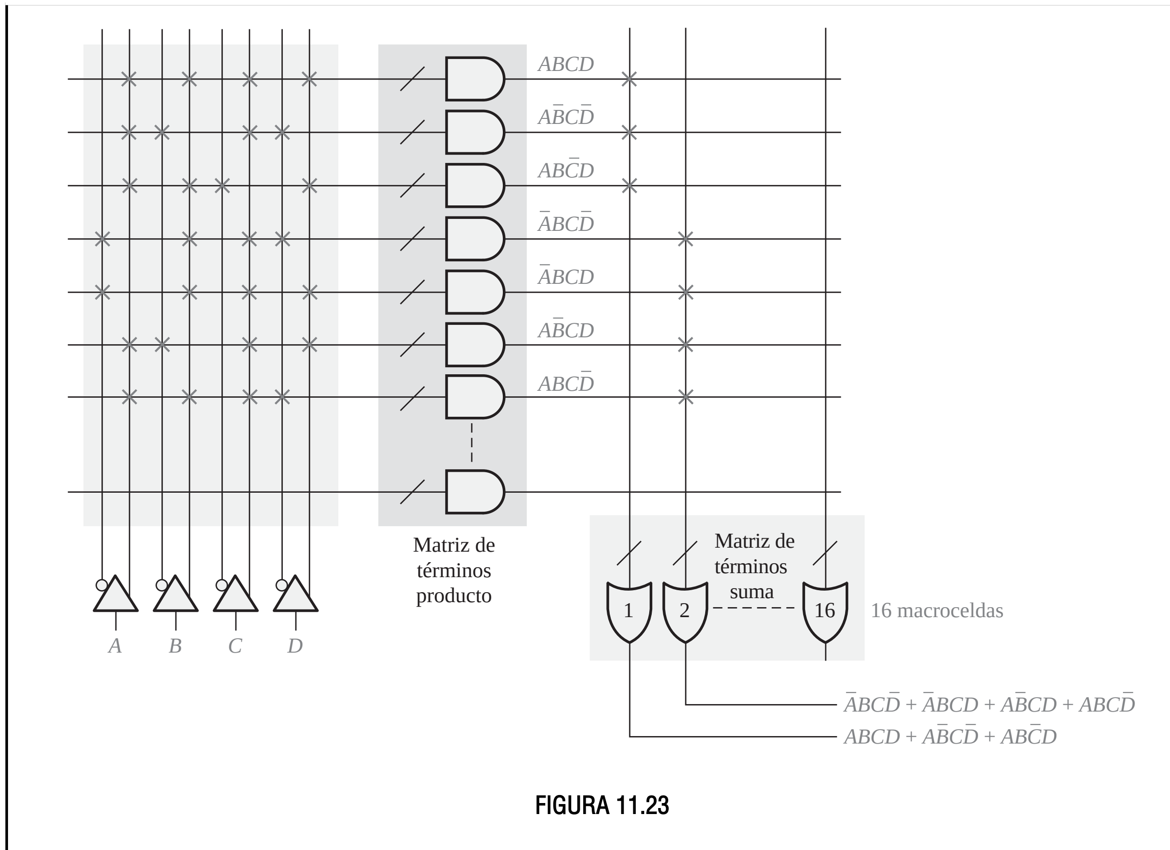


FIGURA 11.23

REVISIÓN DE LA SECCIÓN 11.3

1. ¿Cuál es la diferencia principal entre los dispositivos CPLD de Altera y de Xilinx?
2. Describa una PLA.
3. ¿En qué se diferencia una PAL de una PAL?
4. ¿Qué significa FB?

11.4 MACROCELDAS

Las macroceldas CPLD ya han sido presentadas en las secciones anteriores, dedicadas a los dispositivos de Altera y de Xilinx. Recuerde que una macrocelda puede reconfigurarse, mediante programación, para implementar salidas y entradas de lógica combinacional o de lógica registrada. El término *registrada* hace referencia a la utilización de flip-flops. En esta sección vamos a estudiar la macrocelda típica, incluyendo los modos de operación combinacional y registrado. Aunque la arquitectura de las macroceldas varía entre los distintos dispositivos CPLD, utilizaremos dispositivos representativos como ilustración.

Al finalizar esta sección, el lector deberá ser capaz de:

- Describir la operación de una macrocelda de los dispositivos CPLD MAX 7000 de Altera.
- Describir la operación de una macrocelda de los dispositivos CPLD CoolRunner II de Xilinx.

Los diagramas lógicos utilizan a menudo el símbolo mostrado en la Figura 11.24 para representar un multiplexor. En este caso, el multiplexor tiene dos entradas de datos y una entrada de selección, que permite la selección programable; la entrada de selección no se suele mostrar en los diagramas lógicos.

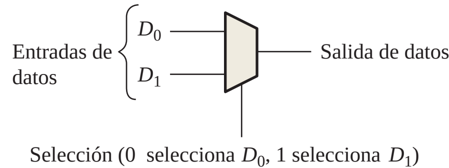


FIGURA 11.24 Símbolo comúnmente utilizado para un multiplexor. Puede tener cualquier número de entradas.

La macrocelda MAX 7000 de Altera

La Figura 11.25 muestra la macrocelda completa, incluyendo el flip-flop (registro). La puerta XOR permite complementar la función suma de productos de la puerta OR, para generar una función en forma de producto de sumas. Un 1 en la entrada superior de la puerta XOR complementa la salida de OR y un 0 permite que la salida de la puerta OR pase sin complementarse (en forma suma de productos). El multiplexor MUX 1 permite seleccionar la salida de la puerta XOR o una entrada procedente de la E/S. El multiplexor 2 puede programarse para seleccionar el reloj global o una señal de reloj basada en un término producto. El multiplexor MUX 3 puede programarse para seleccionar una señal de habilitación para el flip-flop que puede ser un valor ALTO (V_{CC}) o un término producto. El multiplexor MUX 4 permite seleccionar una señal de borrado que puede ser la señal de borrado global o un término producto. MUX 5 se utiliza para puentear el flip-flop y conectar a la E/S la salida de lógica combinacional o la salida registrada. El flip-flop puede programarse como flip-flop D, T (basculación), J-K o S-R.

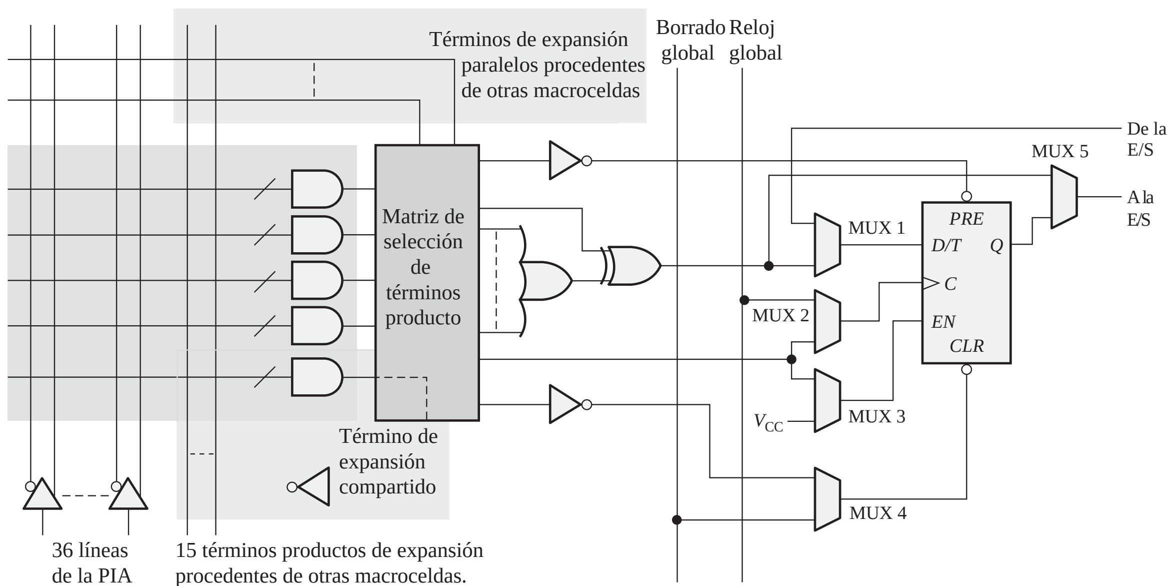


FIGURA 11.25 Una macrocelda de la familia de dispositivos CPLD MAX 7000 de Altera.

El modo combinacional. Cuando se programa una macrocelda para generar una función lógica combinacional suma de productos, los elementos lógicos del camino de los datos son los que se muestran en negro en la Figura 11.26. Como puede verse, sólo se utiliza un multiplexor puenteándose el registro (flip-flop).

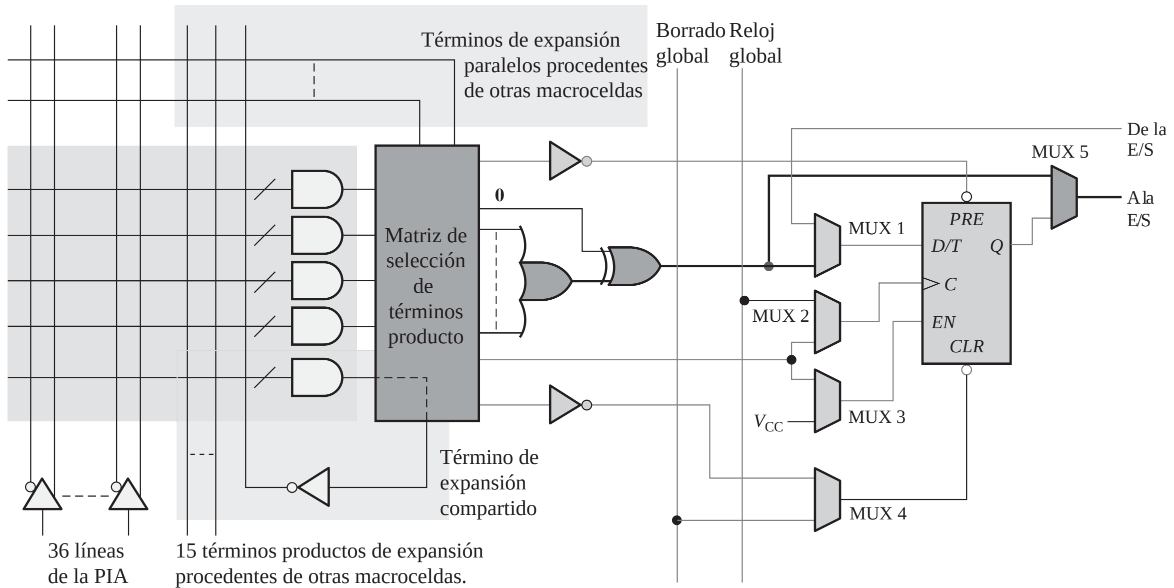


FIGURA 11.26 Una macrocelda configurada para la generación de una función lógica suma de productos. Se indica en negro el camino de los datos.

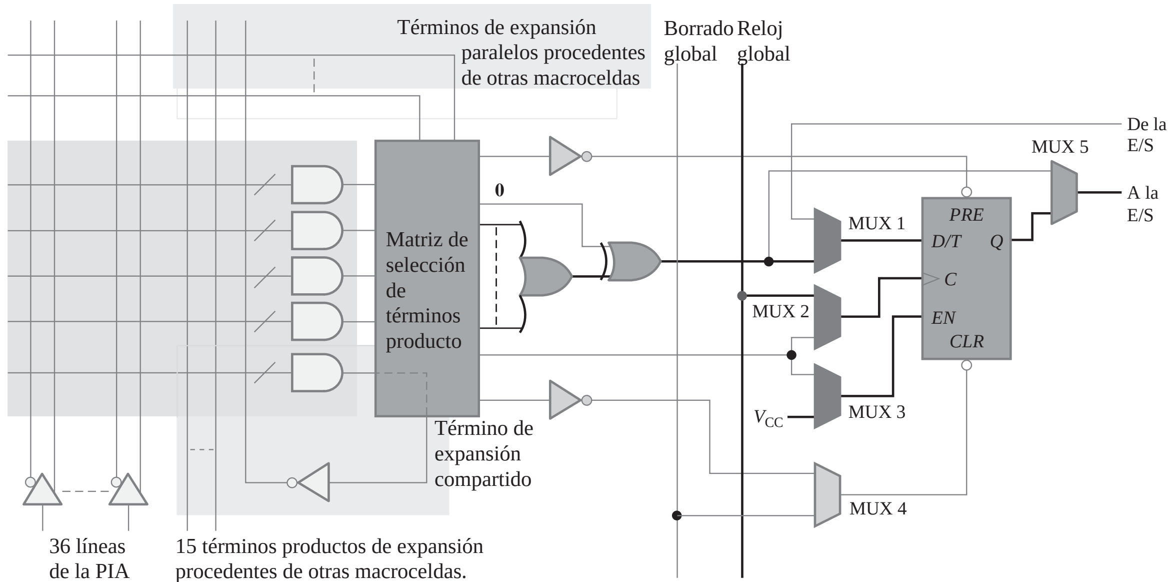


FIGURA 11.27 Una macrocelda configurada para la generación de una función lógica registrada. Se indica en negro el camino de los datos.

El modo registrado. Cuando se programa una macrocelda para el modo registrado, utilizando la salida de lógica combinacional suma de productos para proporcionar la entrada de datos al registro, la cual se enclava mediante el reloj global, los elementos en el camino de los datos son los que se muestran en negro en la Figura 11.27. Como puede ver, se utilizan cuatro multiplexores y el registro (flip-flop) está activo.

La macrocelda CoolRunner II de Xilinx

La macrocelda del CPLD CoolRunner II se ha presentado brevemente en la Sección 11.3. Recuerde que este dispositivo utiliza una arquitectura PLA, en la que tanto la matriz AND como la matriz OR son programables. La Figura 11.28 muestra la lógica completa de esta macrocelda, incluyendo el flip-flop (registro). La puerta OR tiene múltiples entradas procedentes de la matriz AND, como se indica mediante la línea inclinada que atraviesa la línea de entrada.

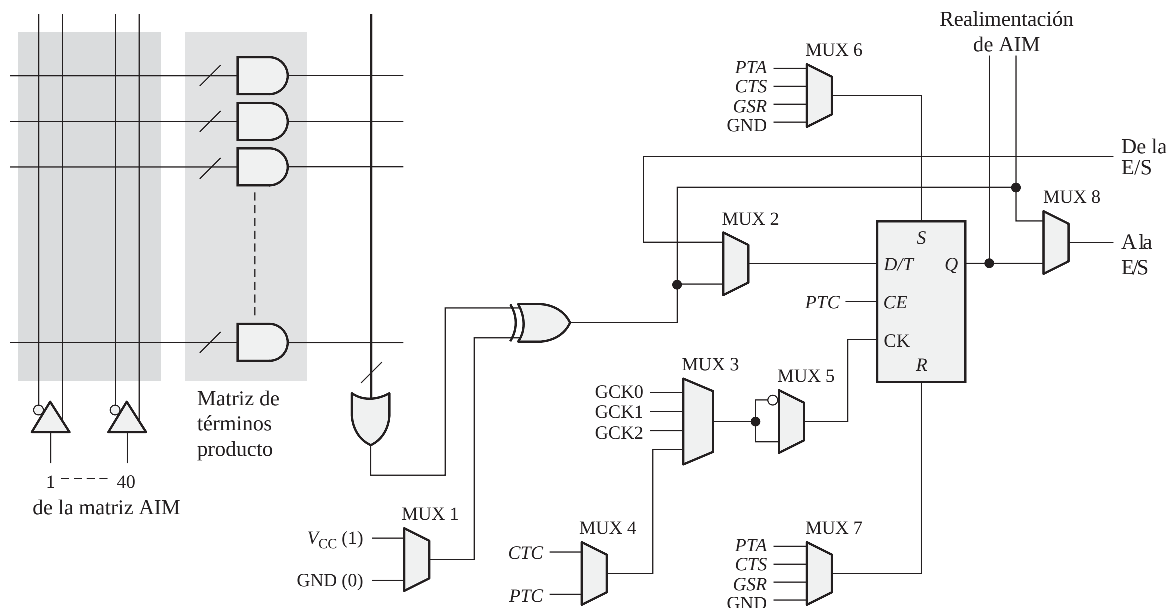


FIGURA 11.28 Una macrocelda de un CPLD CoolRunner II de Xilinx.

La puerta XOR permite complementar la función suma de productos generada en la puerta OR, con el fin de generar una función en forma de producto de sumas. Un 1 en la entrada inferior de la puerta XOR hace que se complemente la salida de la puerta OR mientras que un 0 permite que la salida de la puerta OR pase sin complementar (en forma de suma de productos). El multiplexor MUX 1 permite seleccionar salidas lógicas suma de productos o producto de sumas. El multiplexor MUX 2 permite seleccionar la salida de la puerta XOR o una entrada de E/S. MUX 3 y MUX 4 pueden programarse para seleccionar uno de los relojes globales (GCK0, GCK1 o GCK2) o una señal de reloj basada en un término producto (*CTC* o *PTC*). *CTC* es un término compartido y *PTC* es un término generado localmente. MUX 5 puede programarse para proporcionar cualquiera de las dos polaridades de la señal de reloj. El término producto *PTC* se utiliza para proporcionar una habilitación de reloj al flip-flop. MUX 6 permite seleccionar una de cuatro señales para poner a nivel ALTO el flip-flop. Estas señales son *PTA* (término producto generado localmente), *CTS* (término producto compartido), *GSR* (*global set/reset*, activación/desactivación global) y GND, que es la línea que normalmen-

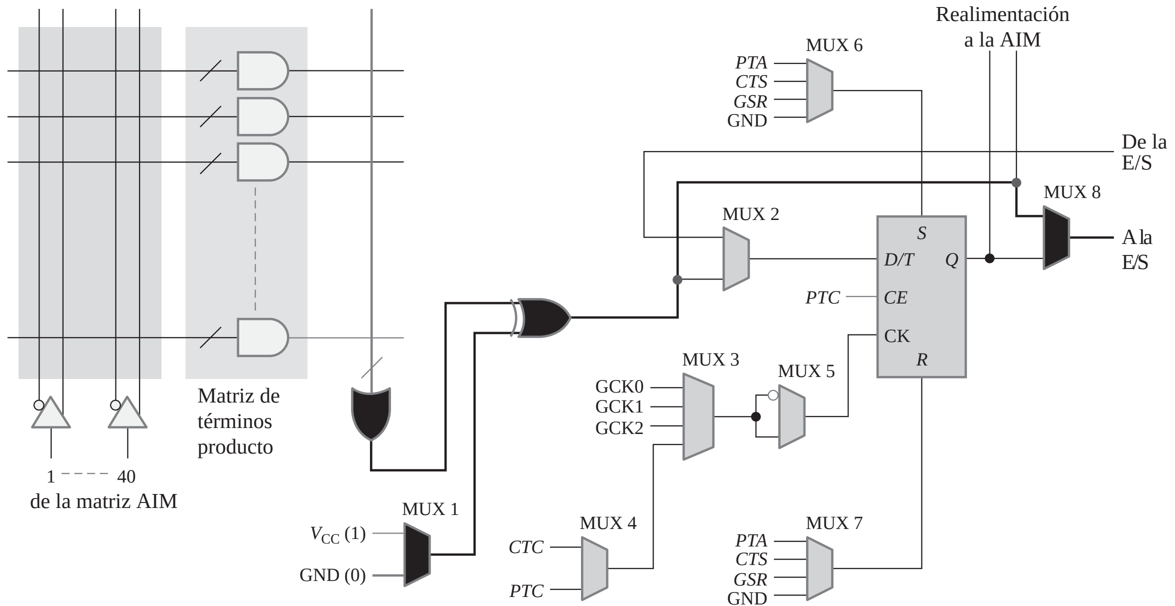


FIGURA 11.29 Una macrocelda configurada para la generación de una función lógica suma de productos. El color negro indica el camino de los datos.

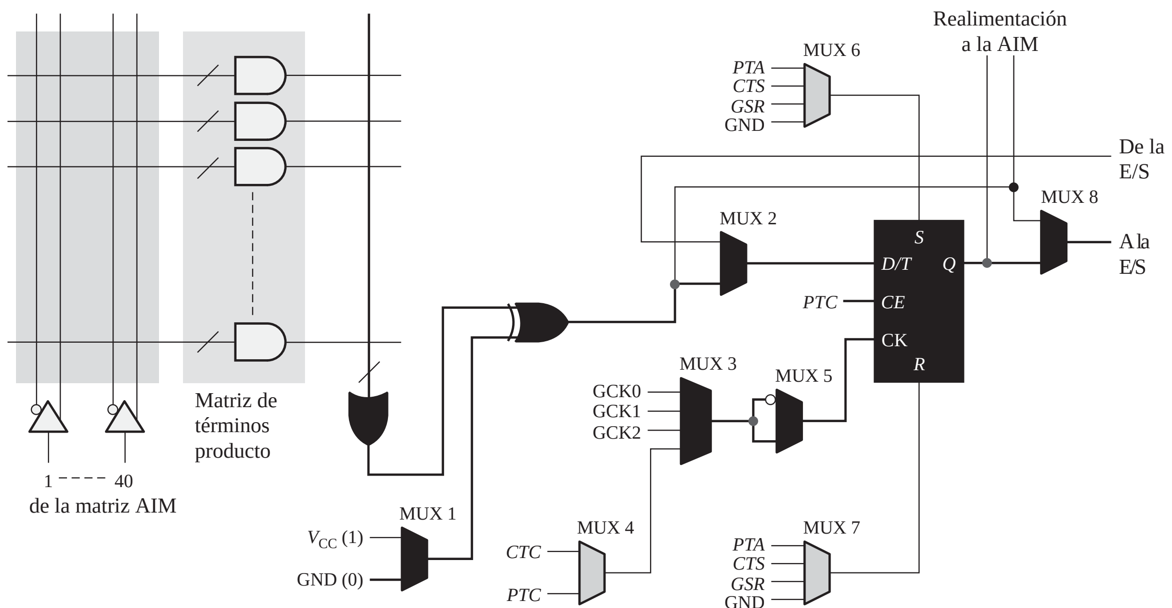


FIGURA 11.30 Una macrocelda configurada para la generación de una función lógica registrada. El color negro indica el camino de los datos.

te se selecciona cuando no se requiere una señal SET activa. MUX 7 proporciona las mismas funciones que MUX 6, pero para borrar o reinicializar el flip-flop. MUX 8 se emplea para puentear el flip-flop y conectar a

la E/S la salida de lógica combinacional o la salida registrada. La salida puede programarse como un flip-flop D, T (basculación), o como *latch*.

El modo combinacional. Cuando se programa una macrocelda para generar una función de lógica combinacional en forma suma de productos, los elementos lógicos que componen el camino de los datos son los que se muestran en negro en la Figura 11.29. Como puede verse sólo se emplean dos multiplexores y el registro (flip-flop) se puentea.

El modo registrado. Cuando se programa una macrocelda para el modo registrado, utilizando la salida de la lógica combinacional de suma de productos para proporcionar la entrada de datos del registro que se enclava mediante uno de los relojes globales, los elementos del camino de datos son los mostrados en negro en la Figura 11.30. Como puede verse, se emplean cinco multiplexores y el registro (flip-flop) está activo.

REVISIÓN DE LA SECCIÓN 11.4

1. Explicar el propósito de la puerta XOR en la macrocelda.
2. ¿Cuáles son los dos modos principales de una macrocelda?
3. ¿A qué hace referencia el término *registrado*?
4. Además de la puerta OR, de la puerta XOR y del flip-flop, ¿qué otro elemento lógico se utiliza comúnmente en una macrocelda?

11.5 LÓGICA PROGRAMABLE: DISPOSITIVOS FPGA

Como ya hemos visto, la arquitectura CPLD clásica está compuesta de bloques lógicos de tipo PAL/GAL o PLA, con interconexiones programables. Básicamente, una FPGA (*Field-Programmable Gate Array*, matriz de puertas programable sobre el terreno) difiere en cuanto a la arquitectura, no utiliza matrices de tipo PAL/PLA y tiene unas densidades mucho mayores que los dispositivos CPLD. Una FPGA típica tiene un número de puertas equivalentes mucho mayor que un dispositivo CPLD típico. Los elementos que implementan las funciones lógicas en las FPGA son, generalmente, mucho más pequeños que en los CPLD, por lo que hay muchos más de esos elementos. Asimismo, en las FPGA, las interconexiones programables están organizadas generalmente según una disposición de filas y columnas.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Describir la estructura básica de una FPGA. ■ Comparar una FPGA con un CPLD. ■ Explicar los bloques LUT. ■ Explicar un dispositivo FPGA basado en SRAM. ■ Definir el concepto de módulo FPGA.

El concepto básico de FPGA ya ha sido presentado en el Capítulo 1. Los tres elementos básicos en una ***FPGA*** son el bloque lógico configurable (CLB, *Configurable Logic Block*), las interconexiones y los bloques de entrada/salida (E/S), como se ilustra en la Figura 11.31. Los bloques CLB de una FPGA no son tan complejos como los bloques LAB o FB de un CPLD, pero suele haber muchos más bloques CLB. Cuando los bloques CLB son relativamente simples, decimos que la arquitectura FPGA es de *granularidad fina*. Cuando los bloques CLB son de mayor tamaño y más complejos, decimos que la arquitectura es de *granularidad gruesa.* Los bloques de E/S situados alrededor del perímetro de la estructura proporcionan un acceso de entrada/salida o bidireccional, individualmente seleccionable, hacia el mundo exterior. La matriz distribuida de interconexiones programables permite interconectar los bloques CLB entre sí y conectarlos a las entradas y a las salidas. Los dispositivos FPGA de gran tamaño pueden tener decenas de miles de bloques CLB, además de memoria y otros recursos.

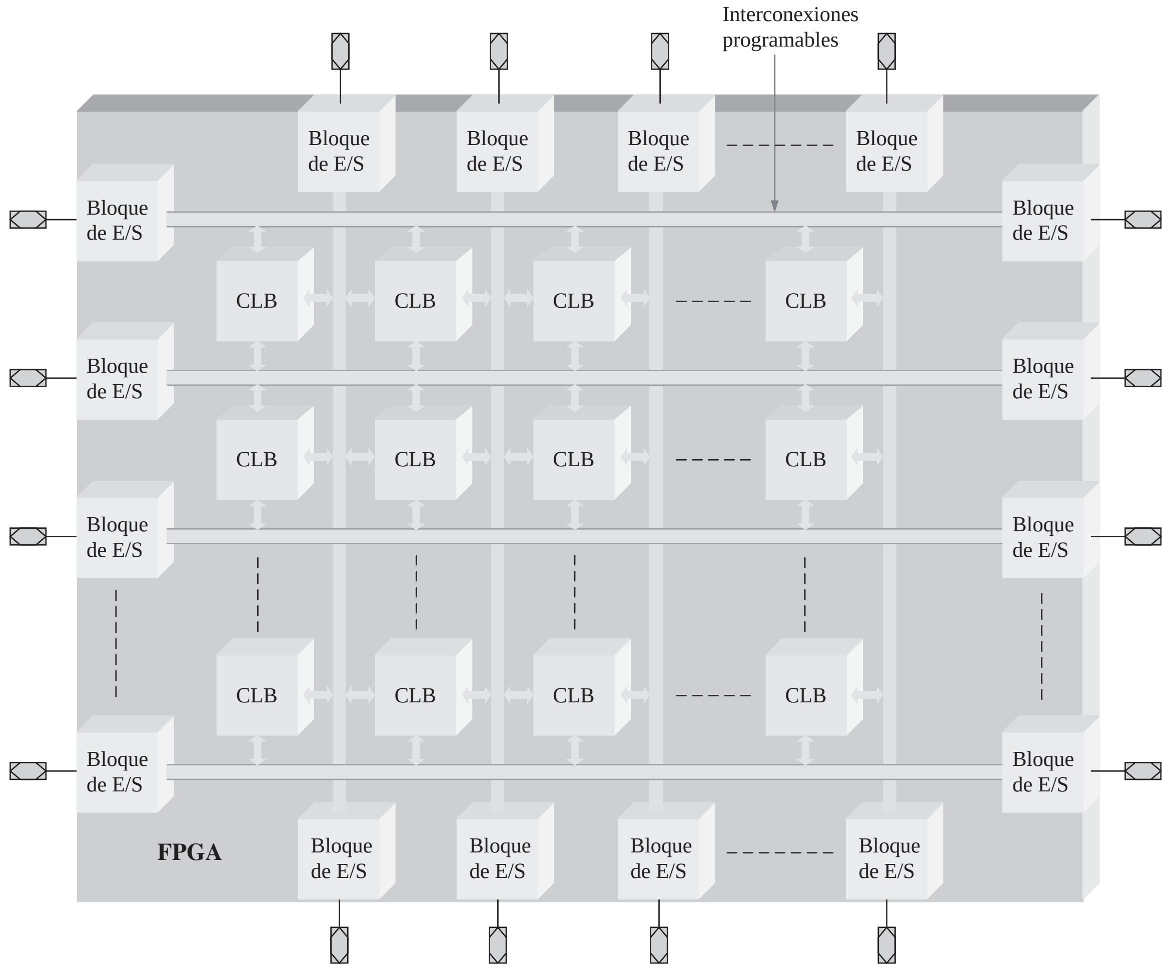


FIGURA 11.31 Estructura básica de una FPGA. Cada CLB es un bloque lógico configurable.

La mayoría de los fabricantes de lógica programable suministran una serie de dispositivos FPGA que varían en cuanto a densidad, consumo de potencia, tensión de alimentación, velocidad y, hasta un cierto punto, también en cuanto a la arquitectura. Las FPGA son reprogramables y utilizan tecnología de proceso SRAM o de antifusibles para implementar las conexiones programables. Las densidades pueden ir desde los centenares de módulos lógicos hasta aproximadamente 180.000 módulos lógicos, en encapsulados de hasta más de 1000 pines. Las tensiones de alimentación continua están comprendidas habitualmente entre 1,2 V y 2,5 V, dependiendo del dispositivo específico.

Bloques lógicos configurables

Normalmente, un bloque lógico de FPGA está compuesto por varios módulos lógicos más pequeños, que son las unidades componentes básicas y que en cierto modo resultan análogos a las macroceldas de un CPLD. La Figura 11.32 muestra los bloques lógicos configurables fundamentales (CLB) dentro de la matriz global de interconexiones programables dispuestas en filas/columnas y que se utilizan para conectar entre sí los bloques lógicos. Cada ***CLB*** está formado por múltiples módulos lógicos más pequeños y por una serie de inter-

conexiones programables locales que se emplean para conectar entre sí los módulos lógicos que componen el CLB.

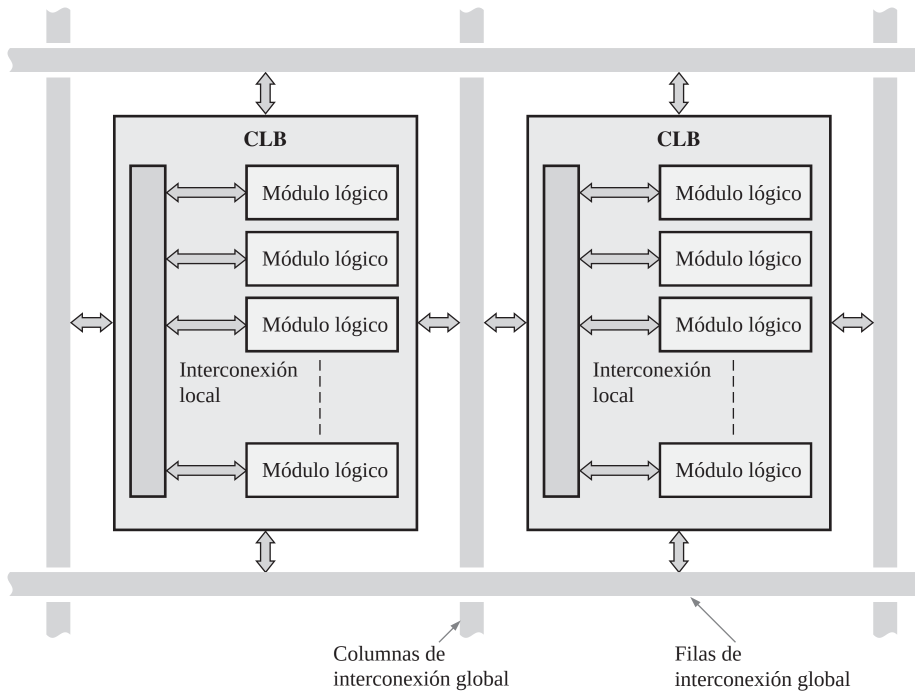


FIGURA 11.32 Bloques lógicos configurables básicos (bloques CLB) dentro de la estructura global de interconexiones programables dispuestas en filas/columnas.

Módulos lógicos. Un módulo lógico de un bloque lógico de una FPGA puede configurarse para implementar lógica combinacional, lógica registrada o una combinación de ambas. Se emplea un flip-flop que forma parte de la lógica asociada para implementar lógica registrada (los flip-flops se han estudiado en el Capítulo 7). En la Figura 11.33 se muestra un diagrama de bloques de un módulo lógico típico basado en LUT. Como ya sabemos, una LUT (*Look-Up Table*) es un tipo de memoria programable que se utiliza para generar funciones lógicas combinacionales suma de productos. La LUT realiza, esencialmente, el mismo trabajo que una PAL o una PLA.

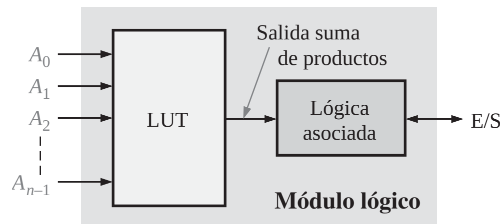


FIGURA 11.33 Diagrama de bloques básico de un módulo lógico de una FPGA.

Generalmente, la organización de una LUT consiste en una serie de 2^n celdas de memoria, siendo n el número de variables de entrada. Por ejemplo, mediante tres entradas se pueden seleccionar hasta ocho celdas

de memoria, por lo que una LUT con tres variables de entrada permite generar una suma de productos con hasta ocho términos. Dentro de las celdas de memoria LUT puede programarse un patrón de 1s y 0s, como se ilustra en la Figura 11.34 para una función suma de productos especificada. Cada 1 significa que el término producto asociado aparecerá en la salida suma de producto mientras que un 0 significa que dicho término producto asociado no aparecerá en la salida suma de productos. La expresión de la salida suma de productos resultante es:

$$\bar{A}_2\bar{A}_1\bar{A}_0 + \bar{A}_2A_1A_0 + A_2\bar{A}_1A_0 + A_2A_1A_0$$

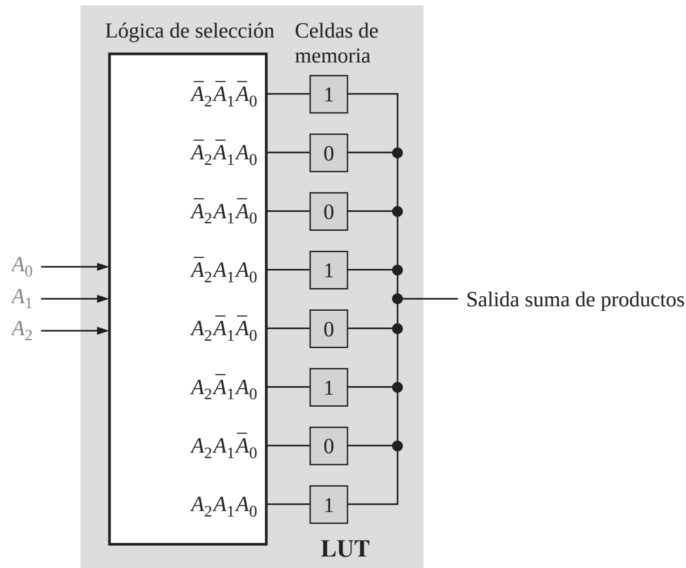


FIGURA 11.34 Concepto básico de una LUT programada para una salida suma de productos concreta.

Dispositivos FPGA basados en SRAM

Los dispositivos FPGA pueden ser no volátiles si están basados en tecnología antifusible, o volátiles, si están basados en tecnología SRAM. El término *volátil* significa que todos los datos programados en los bloques lógicos configurables se pierden cuando se desconecta la alimentación. Por tanto, los dispositivos FPGA basa-

EJEMPLO 11.3

Mostrar una LUT básica de 3 variables para generar la siguiente función suma de productos:

$$A_2A_1\bar{A}_0 + A_2\bar{A}_1\bar{A}_0 + \bar{A}_2A_1A_0 + A_2\bar{A}_1A_0 + \bar{A}_2\bar{A}_1A_0$$

Solución Se almacena un 1 para cada término producto de la expresión suma de productos, como se muestra en la Figura 11.35.

Problema relacionado ¿Cuántas celdas de memoria habría en una LUT con 4 variables de entrada? ¿Cuál sería el máximo número posible de términos producto en la salida suma de productos?

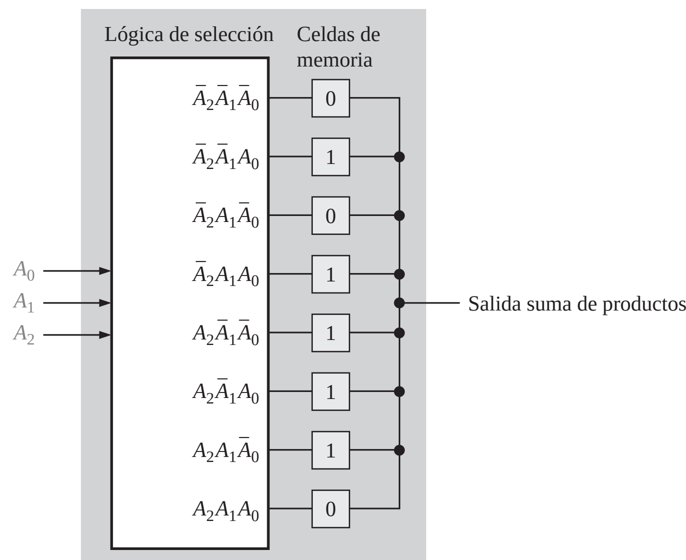


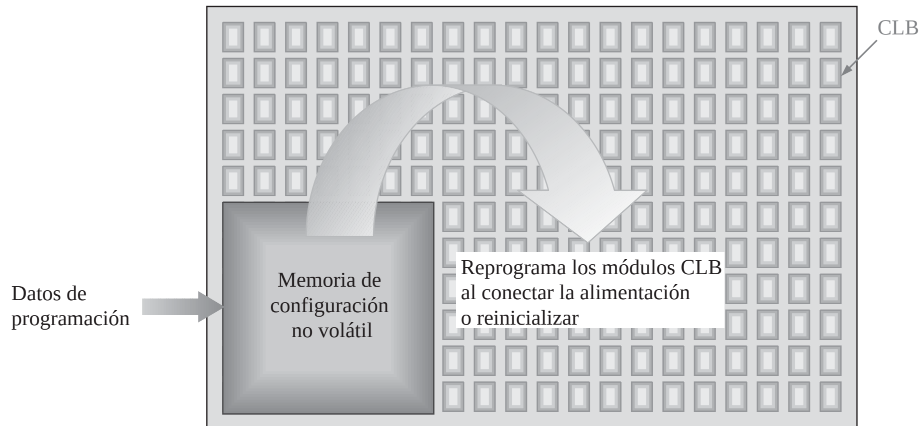
FIGURA 11.35

dos en SRAM o incluyen una memoria de configuración no volátil integrada en el chip para almacenar los datos de programa y reconfigurar el dispositivo cada vez que se aplique la alimentación, o bien utilizan una memoria externa, encargándose un procesador *host* de controlar la transferencia de los datos. El concepto de la memoria interior al chip se ilustra en la Figura 11.36(a), mientras que el concepto de la configuración mediante un procesador *host* se muestra en la parte (b).

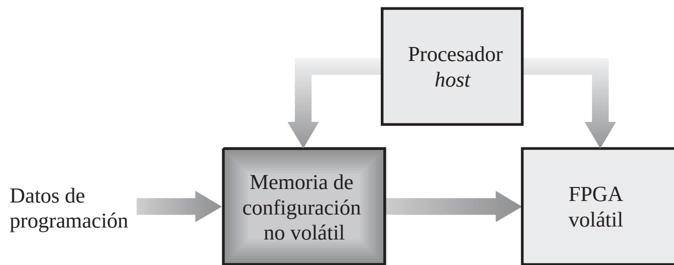
Módulos FPGA

Los dispositivos FPGA, como ya hemos indicado, son esencialmente como "pizarras en blanco", que el usuario final puede programar para implementar cualquier diseño lógico. También hay disponibles dispositivos FPGA que contienen, asimismo, lógica implementada en hardware mediante lo que se denomina un módulo hardware. Un **módulo hardware** es una parte de la lógica dentro de una FPGA que el fabricante incluye para proporcionar una función específica y que no puede reprogramarse. Por ejemplo, si un cliente necesita un pequeño microprocesador como parte del diseño de un sistema, el cliente puede programar ese microprocesador dentro de la FPGA o bien el fabricante puede proporcionar el microprocesador en forma de módulo hardware. Si la función integrada presenta algunas características programables se la conoce con el nombre de **módulo software**. Una ventaja de utilizar módulos hardware es que puede implementarse el mismo diseño empleando una parte mucho más pequeña de la capacidad disponible en la FPGA, por comparación con el caso en que el usuario programara él mismo dicha función; como resultado, se consume menos espacio del chip y se requiere también un menor tiempo de desarrollo por parte del usuario. Asimismo, esas funciones implementadas mediante módulos hardware están convenientemente probadas. La desventaja de los módulos hardware es que las especificaciones se fijan durante la fabricación y el cliente debe utilizar el módulo hardware tal cual se le suministra. Ese módulo no se puede cambiar posteriormente.

Generalmente, hay disponibles módulos hardware para funciones de uso común en los sistemas digitales, como por ejemplo microprocesadores, interfaces estándar de entrada/salida y procesadores digitales de



(a) FGPA volátil con memoria de configuración no volátil interna al chip.



(b) FGPA volátil con un procesador *host* y una memoria integrada en la tarjeta.

FIGURA 11.36 Conceptos básicos relativos a la configuración de una FPGA volátil.

señal. Dentro de una FPGA puede programarse más de un módulo hardware. La Figura 11.37 ilustra el concepto de un módulo hardware rodeado por lógica configurable programada por el usuario. Se trata de un sistema integrado básico, porque el módulo hardware está integrado dentro de la lógica programada por el usuario.

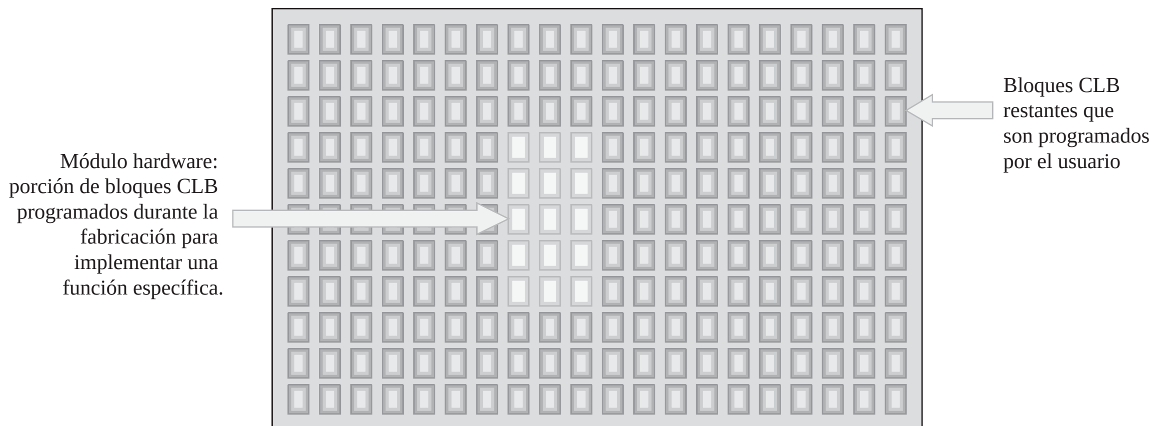


FIGURA 11.37 Concepto básico de un módulo hardware integrado en una FPGA.

Los diseños de módulos hardware suelen desarrollarlos los fabricantes de dispositivos FPGA y son propiedad de dichos fabricantes. Los diseños propiedad del fabricante se denominan ***propiedad intelectual*** (**IP**, *Intellectual Property*). Normalmente, las empresas indican en sus sitios web los tipos de propiedad intelectual disponibles. Parte de esta propiedad intelectual suele ser una mezcla de módulos hardware y módulos software. Un ejemplo sería un procesador que disponga de cierta flexibilidad a la hora de seleccionar y ajustar ciertos parámetros por parte del usuario

Los dispositivos FPGA que contienen procesadores integrados y otras funciones implementadas en forma de módulo hardware y de módulo software se conocen con el nombre de **dispositivos FPGA de plataforma**, porque pueden emplearse para implementar un sistema completo sin necesidad de dispositivos externos de soporte.

REVISIÓN DE LA SECCIÓN 11.5

1. ¿En qué se diferencian una FPGA y un CPLD?
2. ¿Qué significa CLB?
3. Describir una LUT y explicar cuál es su propósito.
4. ¿Cuál es la diferencia entre una interconexión local y una interconexión global en una FPGA?
5. ¿Qué es un módulo FPGA?
6. Describir el término *propiedad intelectual* con relación a los fabricantes de dispositivos FPGA.

11.6 DISPOSITIVOS FPGA DE ALTERA

Altera suministra varias familias de dispositivos FPGA, incluyendo las series Stratix II, Stratix, Cyclone y la familia ACEX. En esta sección, vamos a centrarnos exclusivamente en la familia Stratix II para ilustrar los conceptos, teniendo siempre presente que los otros dispositivos de la familia pueden diferir básicamente en ciertos aspectos de su arquitectura y/o en lo que se refiere a parámetros tales como la densidad, la velocidad y la potencia.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Describir la arquitectura básica de una FPGA típica de la familia Stratix II. ■ Explicar cómo se generan términos producto en las FPGA. ■ Explicar el concepto de funciones integradas.

El bloque LAB (*Logic Array Block*)

Ya hemos mostrado en la Figura 11.31 el diagrama de bloques de una FPGA genérica; la arquitectura de la familia Stratix II y de otras familias de Altera es similar. Tienen la clásica estructura basada en tablas LUT para los módulos lógicos, que Altera denomina módulos lógicos adaptativos (ALM, *Adaptive Logic Module*) y con los que se generan las funciones suma de productos. Asimismo, Altera denomina bloques de matriz lógica (LAB) a los bloques lógicos configurables que se muestran en el dispositivo genérico. La densidad de los dispositivos varía entre algo menos de 2.000 hasta más de 22.000 bloques LAB, dependiendo del dispositivo concreto de la familia y cada bloque LAB tiene ocho módulos ALM. El tamaño de los encapsulados varía entre 341 y 1.173 pines. Los dispositivos más comunes requieren tensiones de alimentación de 1,2 V, 1,5 V y 2,5 V. La familia Stratix II de dispositivos FPGA utiliza una tecnología de proceso basada en SRAM.

La Figura 11.38 muestra un diagrama simplificado de la estructura de un bloque LAB Stratix II. Cada LAB está compuesto de ocho módulos ALM y los distintos bloques LAB se enlazan mediante interconexiones globales de fila y de columna. Las interconexiones locales enlazan los módulos ALM dentro de cada LAB.

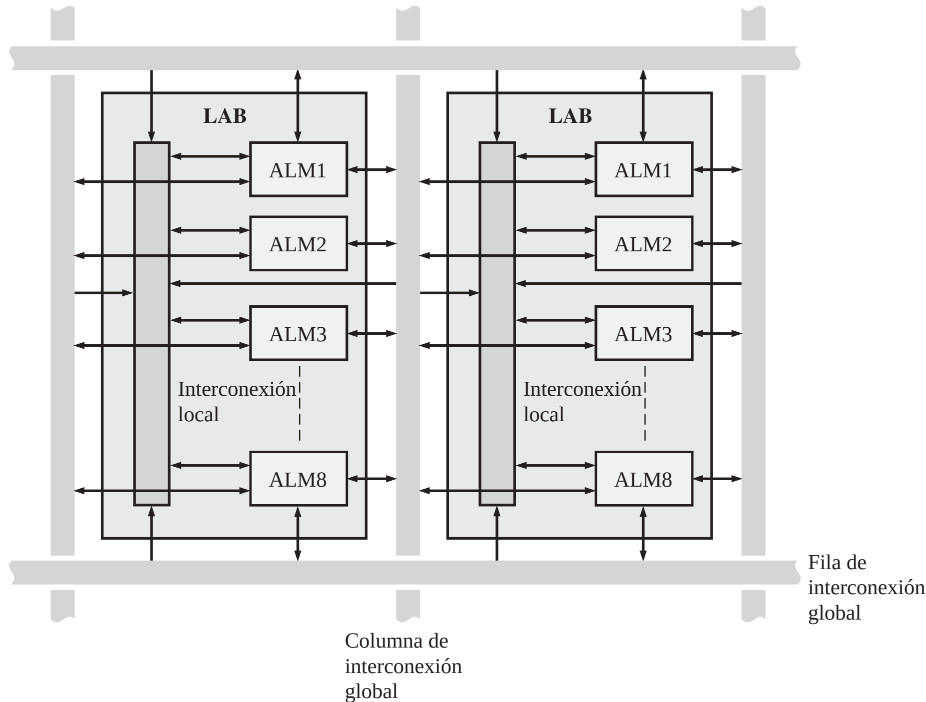


FIGURA 11.38 Diagrama simplificado del bloque LAB (bloque de matriz lógica) de la FPGA Stratix II. Los módulos ALM son módulos lógicos adaptativos.

El módulo ALM (*Adaptive Logic Module*)

El módulo ALM es la unidad básica de diseño en la FPGA Stratix II. Cada ALM contiene una sección de lógica combinacional basada en LUT, junto con otra lógica asociada que puede programarse para obtener dos salida de lógica combinacional o registrada. Asimismo, el ALM tiene lógica de suma, flip-flops y otras secciones de lógica que permiten implementar funciones aritméticas, de recuento y de registros de desplazamiento. En la Figura 11.39 se muestra un diagrama simplificado de un módulo ALM Stratix II.

Modos de operación de un ALM. Un ALM puede programarse para los siguientes modos de operación:

- Modo normal
- Modo LUT ampliado
- Modo aritmético
- Modo aritmético compartido

Además de estos cuatro modos, puede utilizarse un ALM como parte de una cadena de registros con el fin de crear contadores y registros de desplazamiento. En esta sección, vamos a analizar el modo normal y el modo LUT ampliado.

El *modo normal* se utiliza principalmente para generar funciones de lógica combinacional. Un ALM puede implementar una o dos funciones de salida combinacionales gracias a sus dos tablas LUT. En la Figura 11.40 se ilustran cuatro ejemplos de configuraciones LUT. En un ALM, pueden implementar dos funciones suma de productos, cada una de ellas con un máximo de 4 variables sin compartir las entradas. Por ejemplo, se pueden implementar dos funciones de 4 variables, una función de 4 variables y otra de 3 variables, o dos fun-

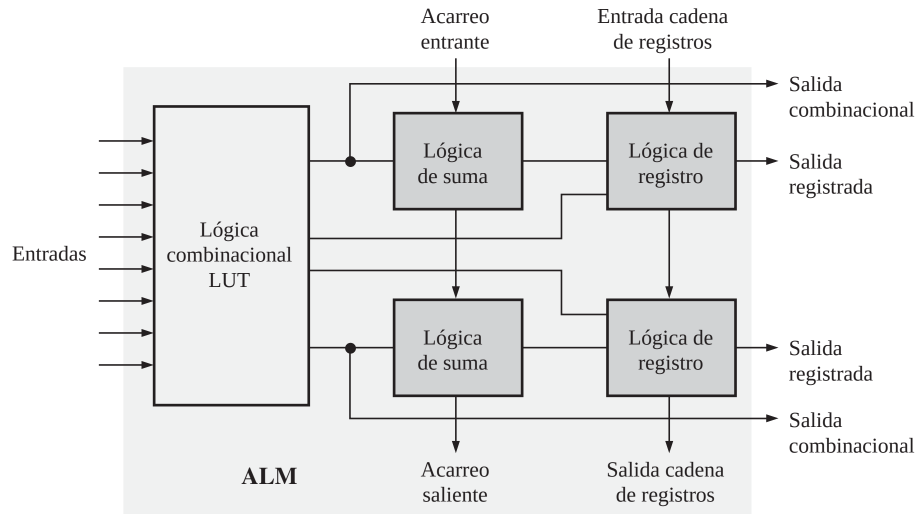


FIGURA 11.39 Diagrama simplificado de un módulo ALM de la FPGA Stratix II.

ciones de 3 variables. Compartiendo las entradas, podemos utilizar cualquier combinación de un total de ocho entradas, hasta un máximo de seis entradas por cada LUT. En el modo normal, estamos limitados a funciones suma de productos de seis variables.

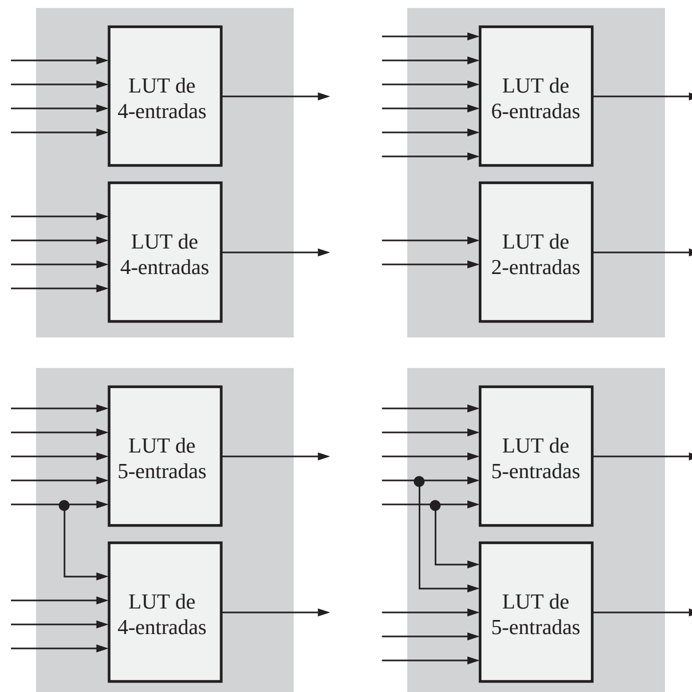


FIGURA 11.40 Ejemplos de posibles configuraciones LUT en un módulo logico adaptativo (ALM) que funcione en el modo normal.

El *modo LUT ampliado* permite realizar la expansión a una función de 7 variables, como se ilustra en la Figura 11.41. El multiplexor constituido por el circuito AND/OR con una entrada complementada forma parte de la lógica dedicada del módulo ALM.

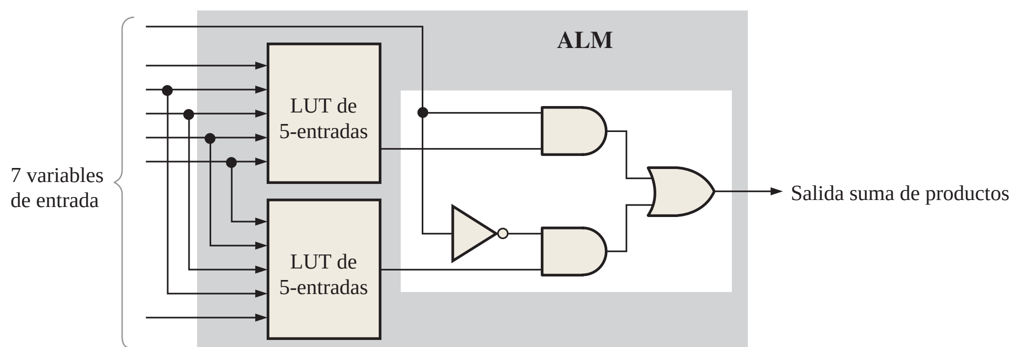


FIGURA 11.41 Expansión de un ALM para generar una función suma de productos de 7 variables en el modo LUT ampliado.

EJEMPLO 11.4

Un ALM de una FPGA Stratix II se configura en el modo LUT ampliado como se muestra en la Figura 11.42. Para las salidas LUT mostradas, determinar la salida final suma de productos.

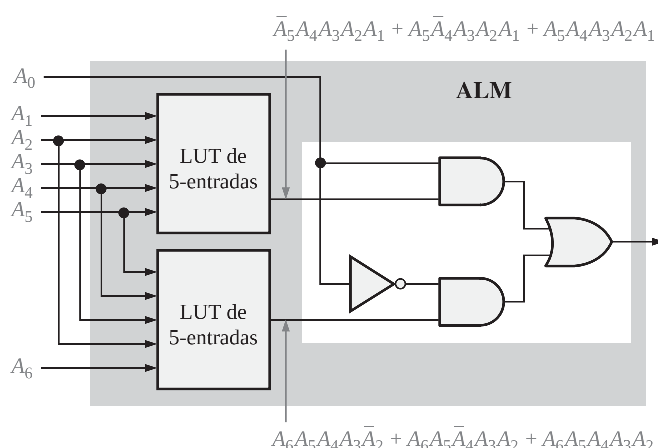


FIGURA 11.42

Solución La expresión suma de productos de salida es la siguiente:

$$\bar{A}_5A_4A_3A_2A_1A_0 + A_5\bar{A}_4A_3A_2A_1A_0 + A_5A_4A_3A_2A_1A_0 + A_6A_5A_4A_3\bar{A}_2\bar{A}_0 + A_6A_5\bar{A}_4A_3A_2\bar{A}_0 + A_6A_5A_4A_3A_2\bar{A}_0$$

Problema relacionado Dibuje una ALM configurada en el modo normal para generar una suma de productos con cinco términos producto provenientes de una LUT y tres términos producto provenientes de la otra.

Funciones integradas

En la Figura 11.43 se muestra un diagrama de bloques general de la FPGA Stratix II. La FPGA contiene funciones de memoria integrada, asi como funciones de procesamiento digital de señal (DSP, *Digital Signal Processing*). Las funciones DSP, como por ejemplo los filtros digitales, se utilizan comúnmente en muchos tipos de sistemas. Como puede ver en el diagrama de bloques, los bloques integrados están distribuidos a todo lo largo de la matriz de interconexión de la FPGA, mientras que los elementos de entrada/salida (IOE, *Input/Output Element*) están situados alrededor del perímetro de la FPGA.

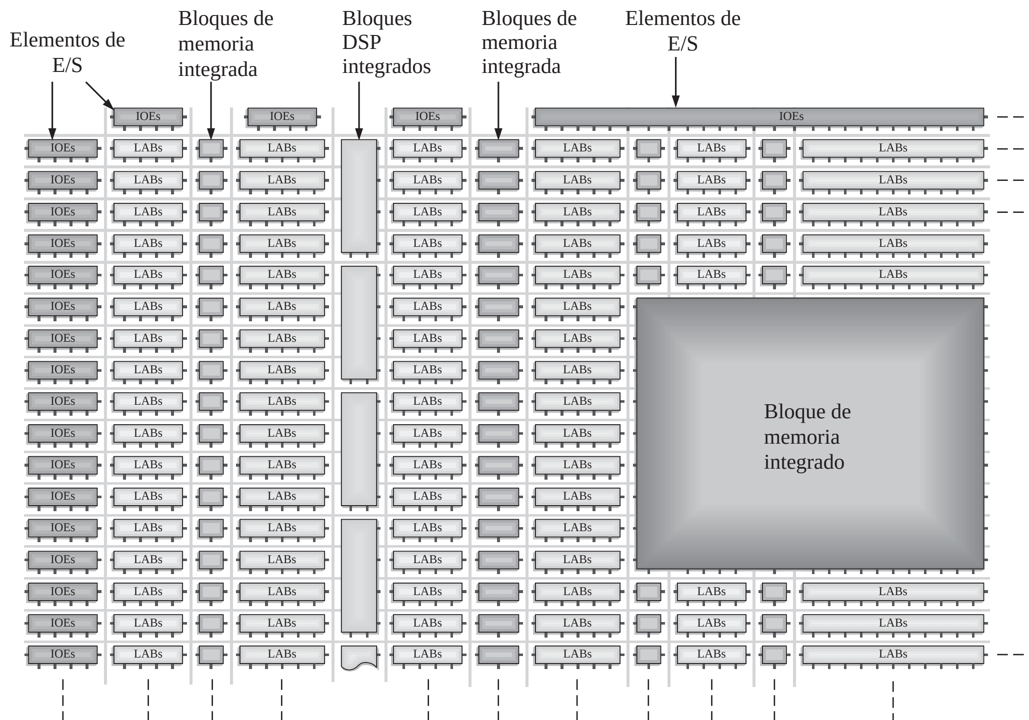


FIGURA 11.43 Diagrama de bloques de la FPGA Stratix II.

REVISIÓN DE LA SECCIÓN 11.6

1. ¿Cuál es la unidad básica de diseño en la FPGA Stratix II?
2. ¿Cuántos módulos ALM hay en un bloque LAB?
3. ¿Qué se utiliza para generar funciones de lógica combinacional en un ALM?
4. ¿Cuántas funciones suma de productos puede generar un ALM?
5. Enumere los dos tipos de funciones integradas que pueden encontrarse en una FPGA Stratix II.

11.7 DISPOSITIVOS FPGA DE XILINX

Xilinx dispone de dos líneas principales de dispositivos FPGA: la Spartan y la Virtex, y existen distintas familias dentro de cada una de las líneas. Como ejemplos podríamos citar las familias Spartan 3 y

Spartan IIE, Virtex-4, Virtex II, Virtex II Pro y Virtex II Pro X. Xilinx designa a las familias Virtex II, Virtex IIPro y Virtex II Pro X como dispositivos FPGA de plataforma porque tienen funciones integradas, como memorias, procesadores, transceptores y otros módulos IP tanto hardware como software. Las diferentes familias de dispositivos FPGA difieren generalmente en cuanto a su densidad y a los parámetros que determinan sus prestaciones. La mayoría de los dispositivos de Xilinx tienen una arquitectura de FGPA tradicional, sin embargo, la familia Virtex II Pro X tiene lo que se denomina una arquitectura basada en ASMBL™ (*Application Specific Modular Block*, bloque modular específico de la aplicación) con más de mil millones de transistores en un mismo dispositivo.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Describir una FPGA típica de la familia Virtex. ■ Explicar la arquitectura básica de la familia Virtex. ■ Explicar cómo se generan términos producto en una FPGA. ■ Describir la arquitectura ASMBL.

Bloques CLB (*Configurable Logic Block*)

El área de lógica configurable (denominada estructura base de la FPGA) en la mayoría de los dispositivos FGPA de Xilinx está dividida en bloques lógicos configurables (CLB), conteniendo cada uno de estos CLB múltiples unidades lógicas básicas denominada celdas lógica (LC, *Logic Cell*). Cada celda lógica está basada en una lógica LUT tradicional de 4-entradas, junto con lógica adicional y un flip-flop. Una LUT de 4-entradas puede generar desde un término producto hasta una función suma de productos con 16 términos distintos. A cada conjunto de dos celdas lógicas idénticas se le denomina *slice* (rodaja). La Figura 11.44 ilustra los distintos niveles de lógica configurable desde la celda lógica al CLB. Las densidades varían de entre unas 2.000 hasta más de 74.000 celdas lógicas en un mismo dispositivo Virtex.

Cadenas suma de productos en cascada

En la Figura 11.45 se muestra una *slice* (dos celdas lógicas) simplificada con lógica de conexión en cascada. Hay un multiplexor (MUX) asociado dentro de la lógica asociada de cada LC que se puede emplear dentro de la cadena de conexión en cascada, así como una puerta OR dedicada dentro de cada *slice.* La Figura 11.45(a) muestra un ejemplo de cómo configurar una *slice* en un CLB como puerta AND, con el fin de generar un término producto de ocho variables. Podemos configurar dos módulos *slices* para generar una función suma de productos con dos términos producto de 8 variables, como se muestra en la parte (b) de la figura. Un CLB completo con cuatro módulos *slice* puede configurarse en una cadena de conexión en cascada con el fin de generar una función suma de productos con cuatro términos producto de 8 variables, como se muestra en la parte (c). Utilizando bloques CLB adicionales puede expandirse aún más la función suma de productos.

EJEMPLO 11.5

Mostrar cómo se puede implementar una puerta AND de 16 entradas en un CLB.

Solución Si se utilizan dos módulos *slices* configurados como se muestra en la Figura 11.46 de la página 721, podremos obtener una puerta AND de 16 entradas.

Problema relacionado Mostrar cómo podrían configurarse dos módulos *slice* en la Figura 11.46 para generar la función suma de productos $A_7A_6A_5A_4 + A_3A_2A_1A_0 + B_7B_6B_5B_4 + B_3B_2B_1B_0$.

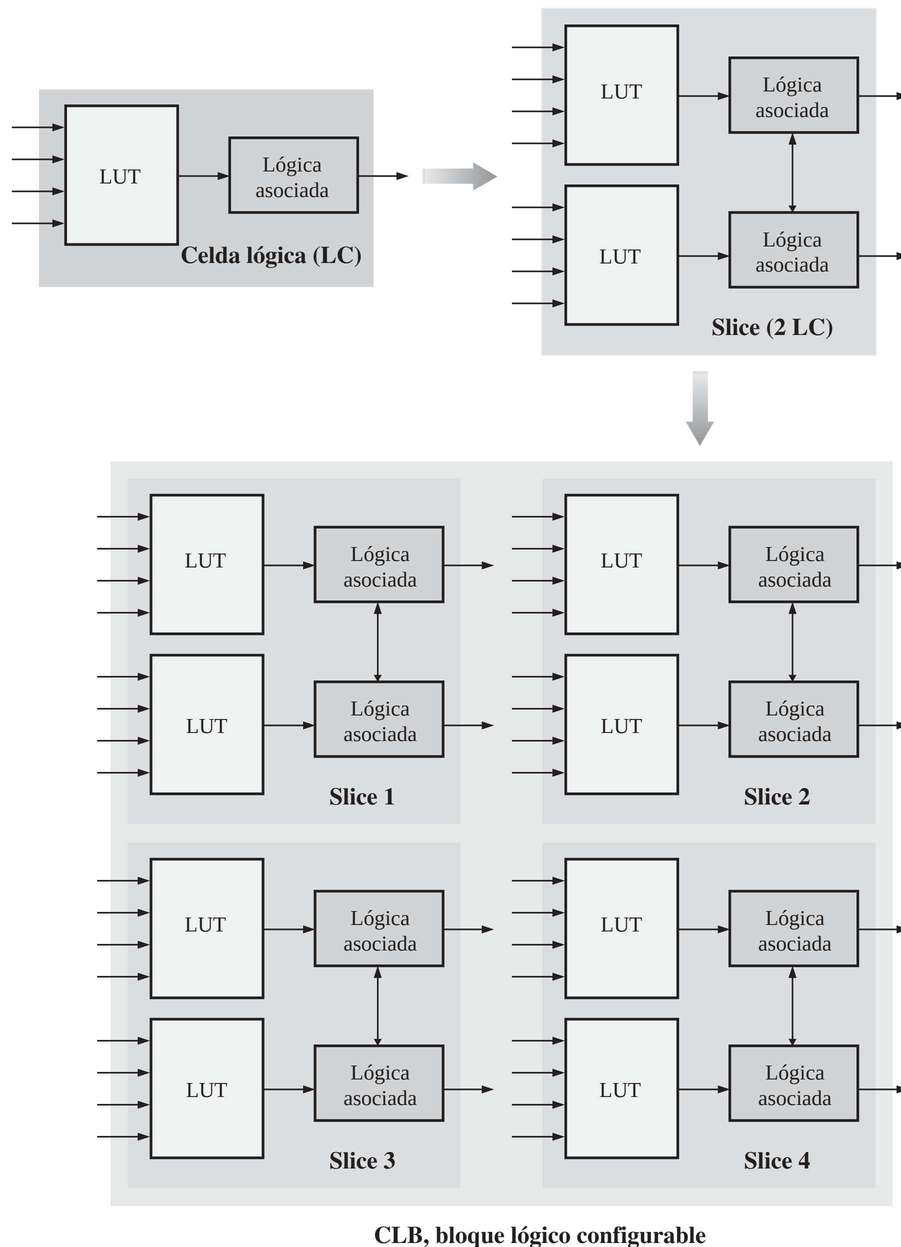


FIGURA 11.44 CLB simplificado en una FPGA Virtex.

Comparación de la arquitectura de una FPGA tradicional y la arquitectura ASMBL

Como hemos visto, la arquitectura tradicional de una FPGA tiene la forma de una matriz de bloques lógicos (CLB o LAB) rodeada por celdas de E/S configurables. La cantidad de lógica configurable (bloques CLB) en una FPGA depende del número de elementos de E/S que puedan situarse físicamente alrededor del perímetro.

Cuando se necesitan módulos IP, tales como un DSP o un bloque de memoria integrada, es necesario sacrificar parte de la lógica configurable; asimismo puede que en algún punto sean necesarias líneas de E/S adicionales. A medida que se añaden más módulos IP, el tamaño físico de la FPGA debe incrementarse para mantener la lógica configurable necesaria e incrementar el número de líneas de E/S. Este concepto general se ilustra en la Figura 11.47.

Cuanto más compleja sea la lógica en una FPGA, más líneas de E/S se requerirán. La relación de dependencia entre la lógica y las líneas de E/S provoca un incremento en el tamaño y en el coste de los chips.

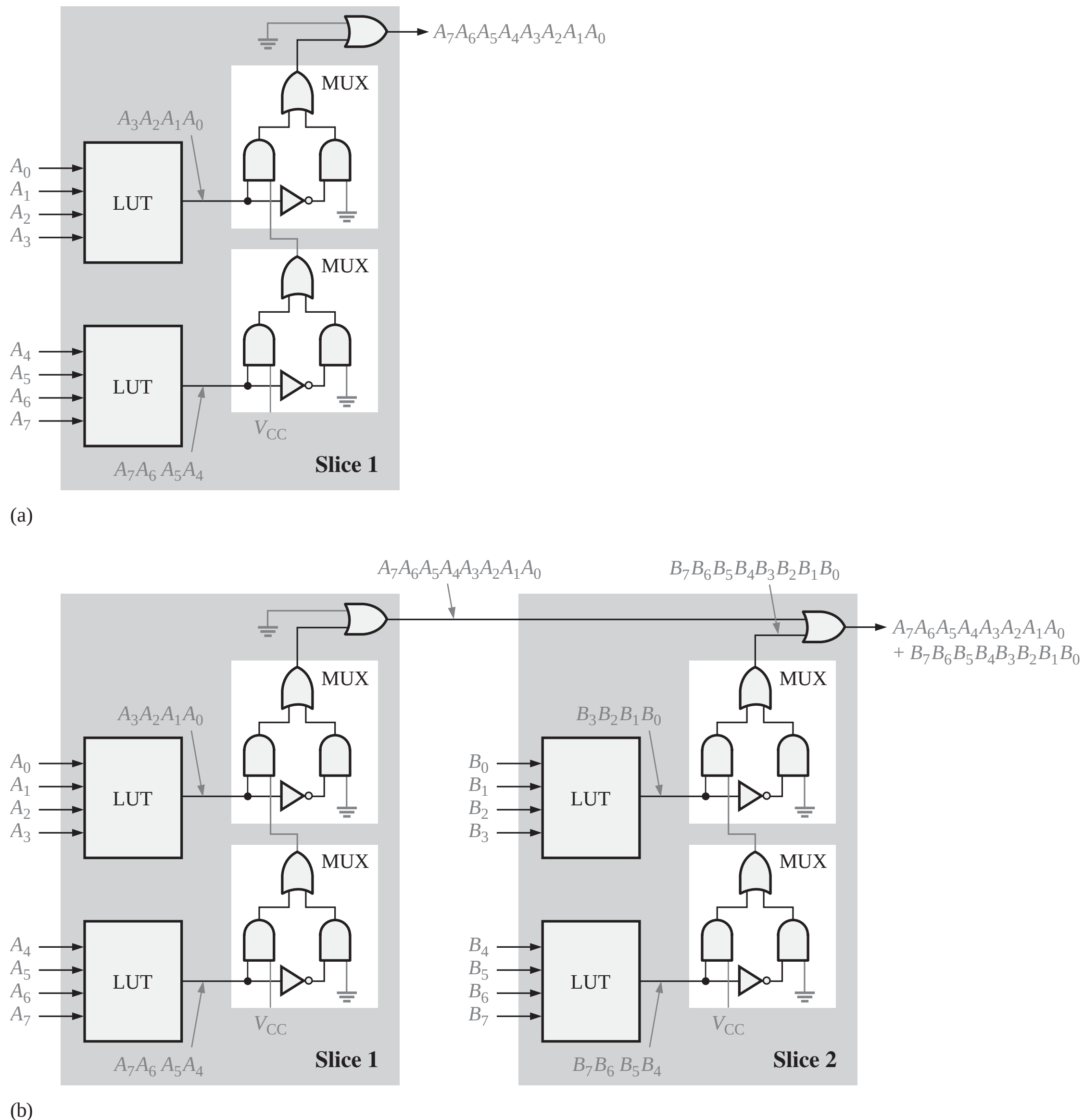


FIGURA 11.45 Ejemplo de utilización de conexiones en cascada para la expansión de una función suma de productos. (*Continúa*)

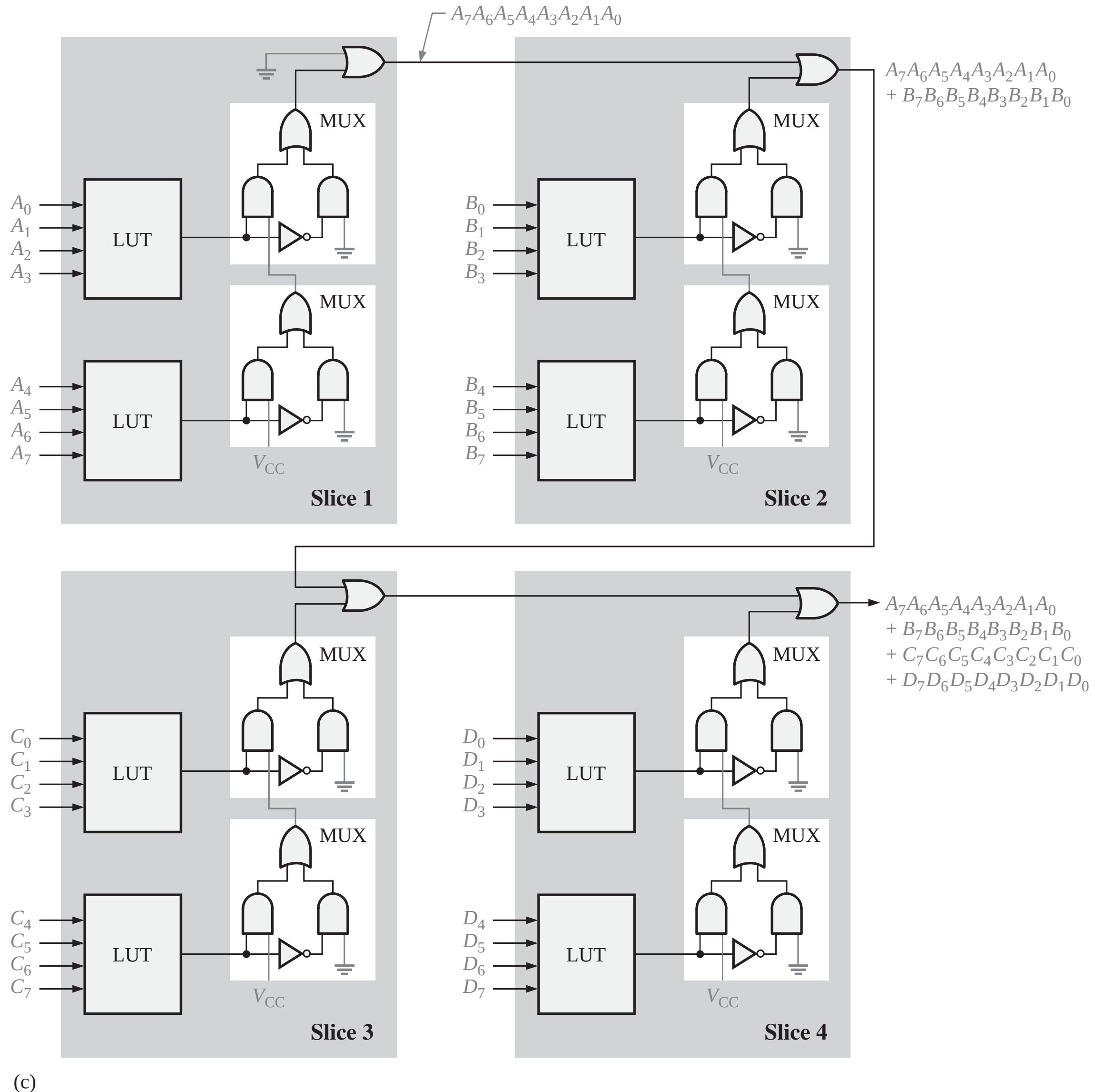


FIGURA 11.45 (*Continuación*)

Asimismo, otro problema con los dispositivos FPGA de plataforma es que cuando se necesitan funciones adicionales integradas en forma de módulos IP, puede que sea preciso realizar un rediseño total o parcial del chip, lo cual es un proceso muy costoso.

La arquitectura ASMBL. Xilinx ha adoptado un enfoque flexible de implementación de los dispositivos FPGA de plataforma con la familia Virtex II Pro X, con el fin de resolver algunas de las limitaciones de la arquitectura tradicional. La arquitectura ASMBL es una estructura basada en columnas, en lugar de estar basada en la estructura tradicional de filas/columnas. Las líneas de E/S están distribuidas por todo el chip, en lugar de estar posicionadas alrededor del perímetro, por lo que puede incrementarse su número sin que aumente el tamaño

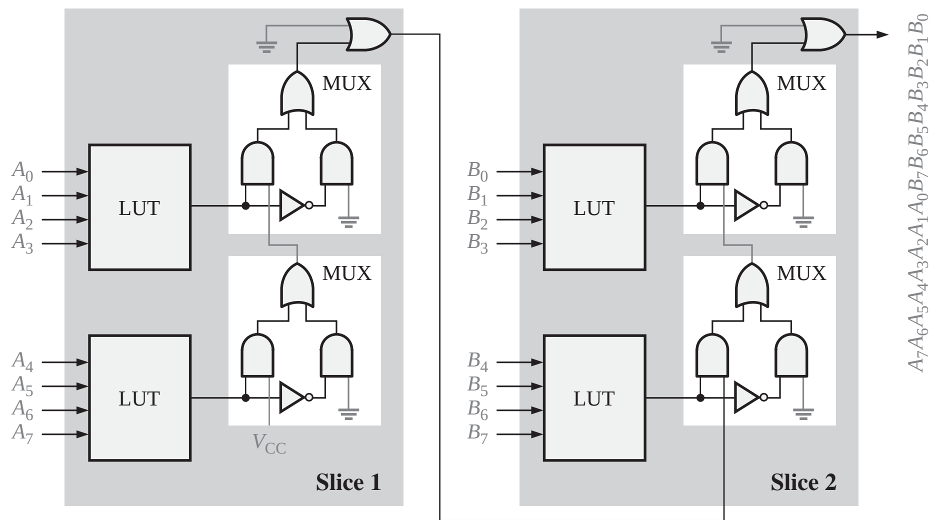
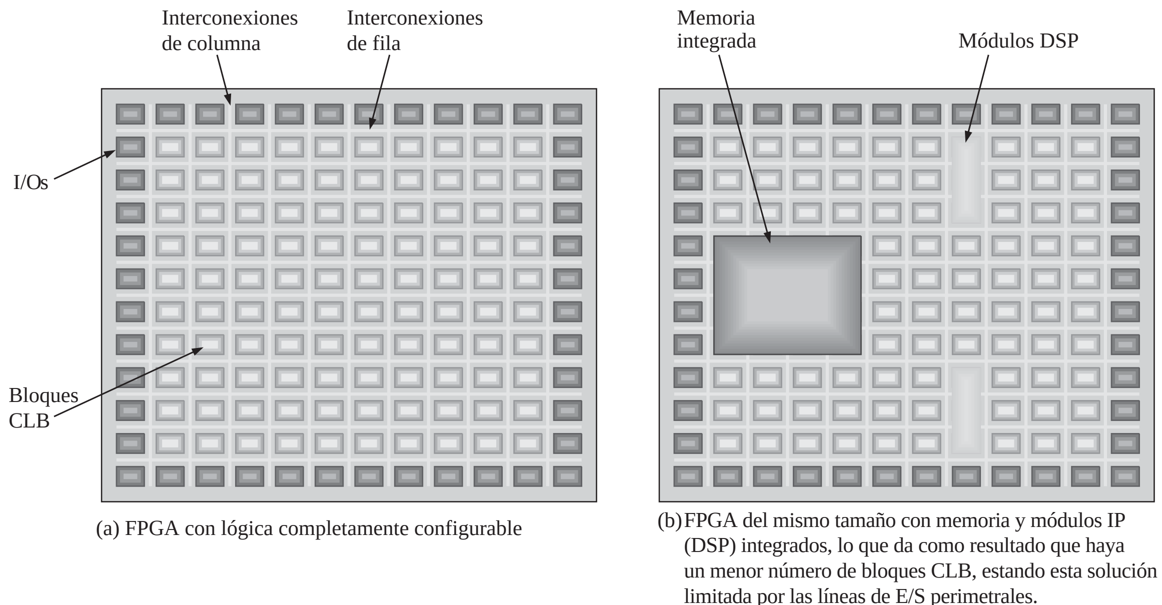


FIGURA 11.46 Implementación de una puerta AND de 16 entradas para generar un término producto con dieciseis variables. (Ejemplo 11.5)

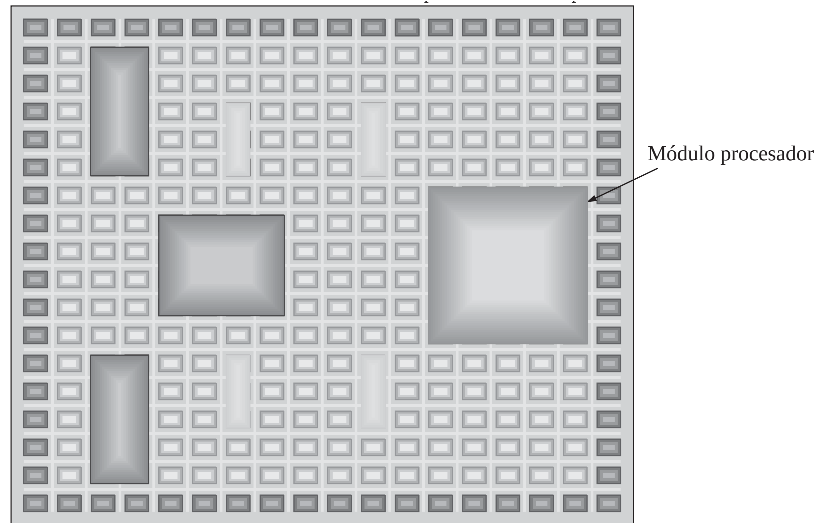
del chip. Cada columna es, esencialmente, una banda de lógica que puede sustituirse por otro tipo de banda lógica sin rediseñar en gran medida la disposición física el chip. Ejemplos de tipos de bandas lógicas son los bloques lógicos configurables (CLB), los bloques de E/S (IOB), la memoria y los módulos IP hardware y software, como por ejemplo los módulos DSP y de procesador.



(a) FPGA con lógica completamente configurable

(b) FPGA del mismo tamaño con memoria y módulos IP (DSP) integrados, lo que da como resultado que haya un menor número de bloques CLB, estando esta solución limitada por las líneas de E/S perimetrales.

FIGURA 11.47 La adición de funciones IP integradas (memoria, DSP y procesador) hace que se disponga de menos lógica configurable y/o que el tamaño físico del chip sea mayor, debido al mayor número de E/S. (*Continúa*)



(c) Una FPGA con más memoria integrada, con módulos DSP adicionales y con un módulo de procesador requerirá en algún momento que se aumente el tamaño físico.

FIGURA 11.47 (*Continuación*)

Pueden mezclarse diversos tipos de banda lógica para satisfacer requisitos específicos de la aplicación. Por ejemplo, en la configuración más sencilla, podríamos tener una mezcla de bandas CLB y bandas de bloques de E/S, como se ilustra en la Figura 11.48(a). Dependiendo de los requisitos se puede emplear un número mayor o menor de cada uno de estos dos tipos. Si hace falta más memoria, podemos sustituir una o más bandas CLB, como se indica en la parte (b). Si el área de aplicación específica es el procesamiento digital de la señal, pueden añadirse módulos IP de DSP con una mezcla de memoria, como se muestra en la parte (c) de la figura. La parte (d) muestra la adición de módulos de procesador.

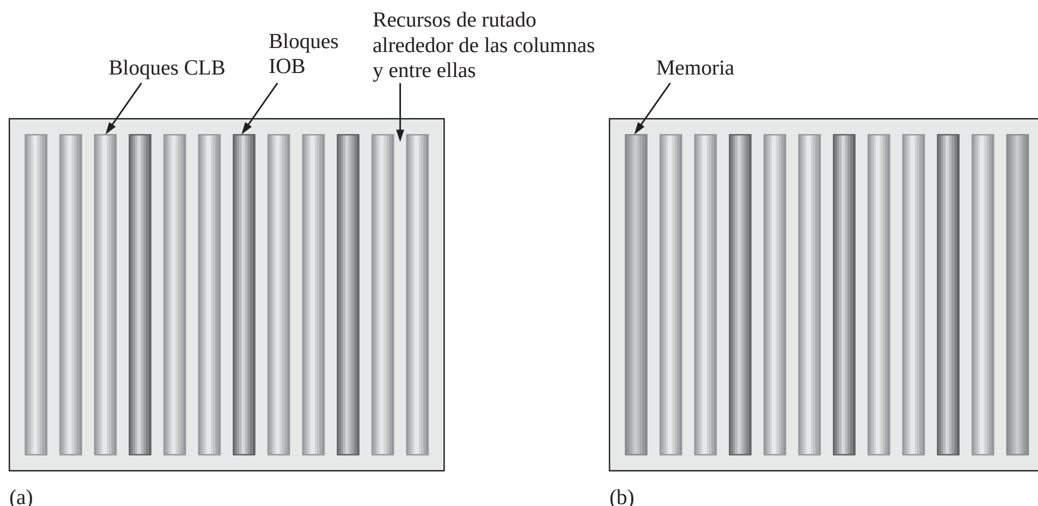


FIGURA 11.48 Concepto básico de la arquitectura ASMBL para dispositivos FPGA de plataforma. (*Continúa*)

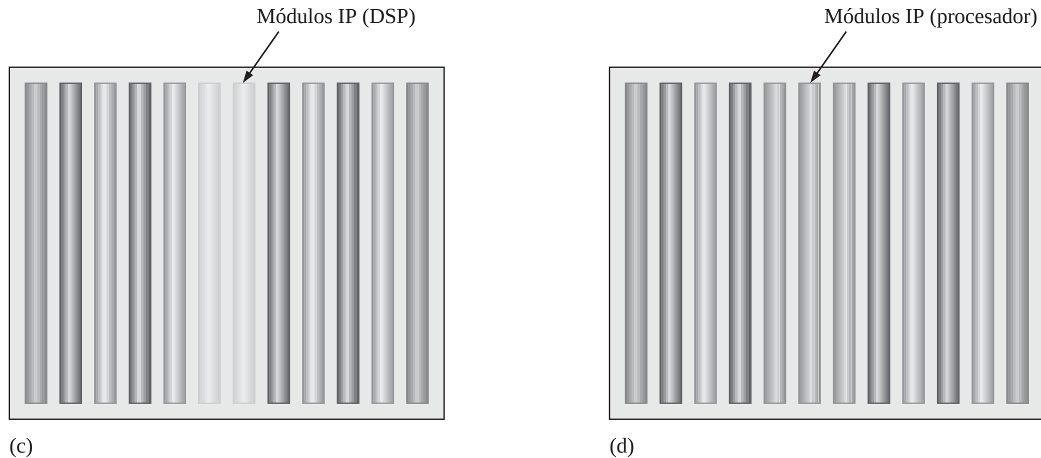


FIGURA 11.48 (*Continuación*)

REVISIÓN DE LA SECCIÓN 11.7

1. ¿Cómo está compuesto un CLB en una FPGA de Xilinx?
2. ¿Cómo está compuesta una celda lógica (LC)?
3. Describir un módulo *slice* de una FPGA de Xilinx.
4. ¿Qué es una cadena de suma de productos de conexión en cascada?
5. ¿Qué quiere decir ASMBL?

11.8 SOFTWARE DE LÓGICA PROGRAMABLE

Para que sea útil, la lógica programable debe permitir combinar componentes tanto hardware como software en una misma unidad funcional. Todos los fabricantes de dispositivos SPLD, CPLD y FPGA proporcionan soporte software para cada dispositivo hardware. Estos paquetes de software se encuentran dentro una categoría de software que se conoce con el nombre de CAD (*Computer Aided Design*, diseño asistido por computadora). En el Capítulo 1 se presentó la programación de dispositivos PLD y se amplió en el Capítulo 3. En esta sección, se presenta de una forma genérica el software de la lógica programable.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Explicar el proceso de programación en términos del flujo de diseño. ■ Describir la fase de introducción del diseño. ■ Describir la fase de simulación funcional. ■ Describir la fase de síntesis. ■ Describir la fase de implementación. ■ Describir la fase de simulación de temporización. ■ Describir la fase de descarga.

El proceso de programación se denomina generalmente ***flujo de diseño***. En la Figura 11.49 se muestra un diagrama básico del flujo de diseño para la implementación de un diseño lógico en un dispositivo programable. La mayoría de los paquetes software específicos incorporan todos estos elementos de una forma u otra y se encargan de realizar el procesamiento de manera automática. El dispositivo que se está programando se denomina usualmente dispositivo de destino.

Son necesarios cuatro elementos para empezar a programar un dispositivo: una computadora, un software de desarrollo, un dispositivo lógico programable (SPLD, CPLD o FPGA) y una forma de conectar el

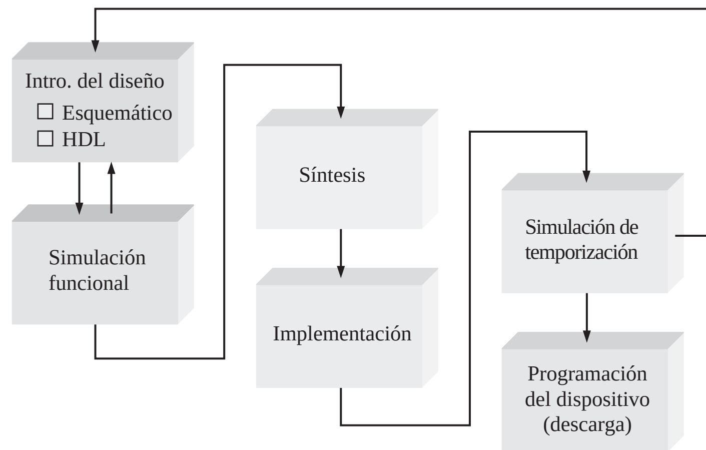
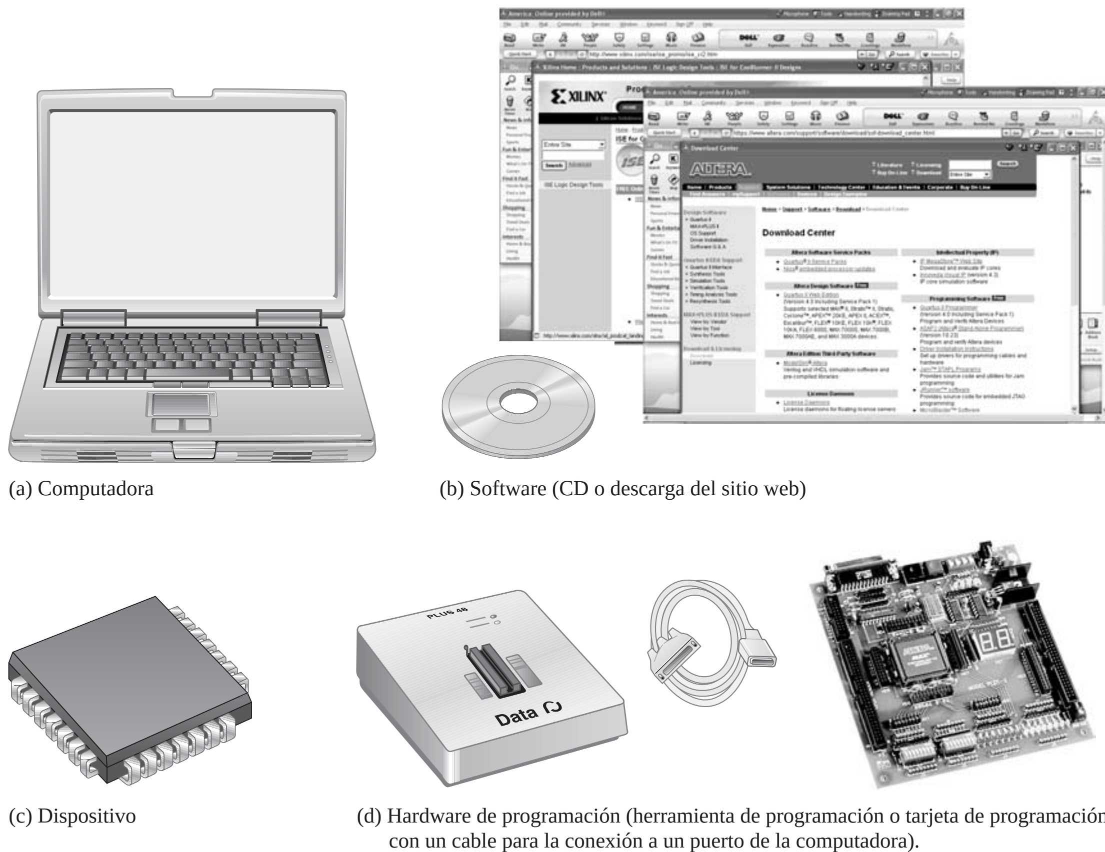


FIGURA 11.49 Diagrama general de flujo de diseño para la programación de un dispositivo SPLD, CPLD o FPGA.



(a) Computadora

(b) Software (CD o descarga del sitio web)

(c) Dispositivo

(d) Hardware de programación (herramienta de programación o tarjeta de programación con un cable para la conexión a un puerto de la computadora).

FIGURA 11.50 Elementos esenciales para programar un dispositivo SPLD, CPLD o FPGA.

dispositivo a la computadora. Estos elementos básicos se ilustran en la Figura 11.50. En la parte (a) se muestra una computadora que cumple los requisitos de sistema para el software concreto que se está utilizando. La

parte (b) muestra el software adquirido bien a través de un CD o como descarga del sitio web del fabricante. La mayoría de los fabricantes proporcionan software gratuito que puede descargarse y utilizarse durante un tiempo limitado. La parte (c) muestra un dispositivo lógico programable. La parte (d) muestra dos formas de conectar físicamente el dispositivo a la computadora a través de cable, utilizando una herramienta de programación en la que se inserta el dispositivo o la tarjeta de desarrollo en la que se ha montado el dispositivo. Después de haber instalado el software en la computadora, es preciso familiarizarse con las herramientas software concretas antes de tratar de conectar y programar un dispositivo. Este proceso de aprendizaje requerirá un considerable tiempo y esfuerzo.

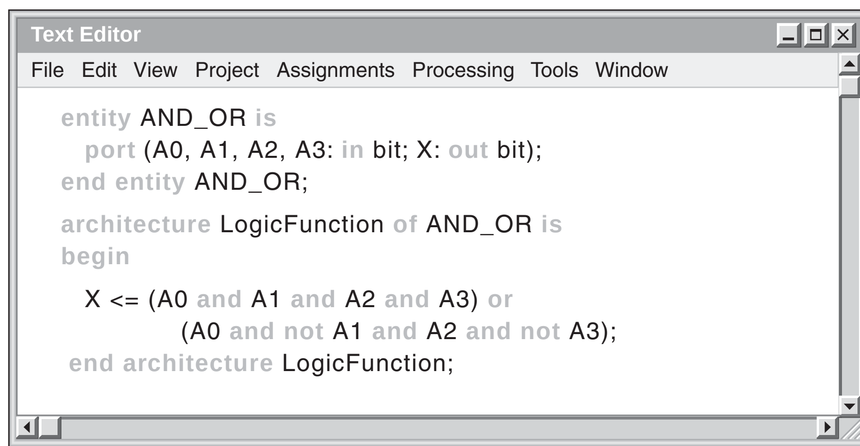
Introducción del diseño

Suponga que dispone de un diseño de circuito lógico que quiere implementar en un dispositivo programable. Podemos introducir el diseño en la computadora utilizando dos formas básicas: ***introducción de esquemáticos*** o ***introducción de textos***. Para poder emplear el método de la introducción de texto, deberá estar familiarizado con un lenguaje de descripción de alto nivel (HDL), como por ejemplo VHDL, Verilog, ABEL o AHDL. La mayoría de los fabricantes de lógica programable proporcionar paquetes de software que soportan VHDL y Verilog, porque se trata de lenguajes HDL estándar. Algunos fabricantes soportan también ABEL, AHDL y otros lenguajes HDL propietario. La introducción de esquemáticos nos permite básicamente colocar en pantalla símbolos de puertas lógicas y otras funciones lógicas de biblioteca y conectarlos según se requiera en el diseño. No hace falta conocer un lenguaje HDL para realizar la introducción de esquemáticos. La Figura 11.51 ilustra genéricamente estos dos tipos de introducción del diseño, para un circuito AND-OR simple.

Construcción de un esquemático lógico. Cuando se introduce un circuito lógico completo a través de la pantalla se denomina esquemático "plano". Puede resultar difícil encajar en la pantalla otros circuitos lógicos más complejos, así que normalmente introduciremos estos circuitos lógicos en segmentos guardando cada segmento en forma de un símbolo de bloque y luego conectando los distintos símbolos de bloque para formar el circuito completo. Esta técnica se denomina diseño jerárquico.

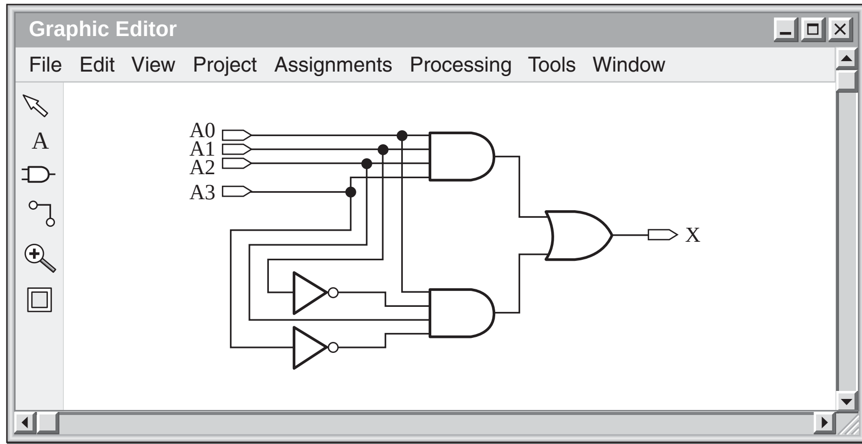
Como ejemplo vamos a suponer que nos hace falta un circuito que genere la siguiente expresión suma de productos:

$$Z = (A_3A_2A_1A_0 + \overline{A}_3A_2\overline{A}_1A_0) + (A_3\overline{A}_2\overline{A}_1A_0 + A_3\overline{A}_2A_1\overline{A}_0 + \overline{A}_3A_2A_1\overline{A}_0)$$



(a) Introducción de texto usando VHDL para describir un circuito lógico AND-OR.

FIGURA 11.51 Ejemplos de pantallas de introducción de texto y de esquemáticos. (*Continúa*).

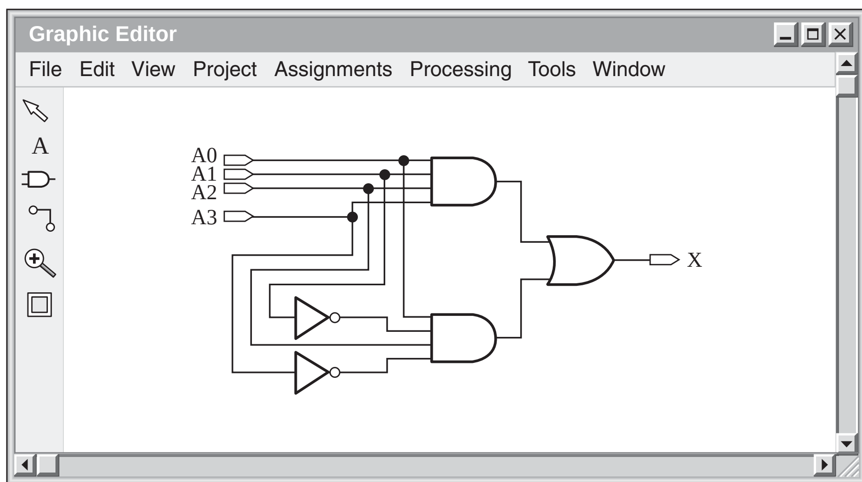


(b) Introducción del esquemático del mismo circuito lógico AND-OR de la parte (a).

FIGURA 11.51 (*Continuación*).

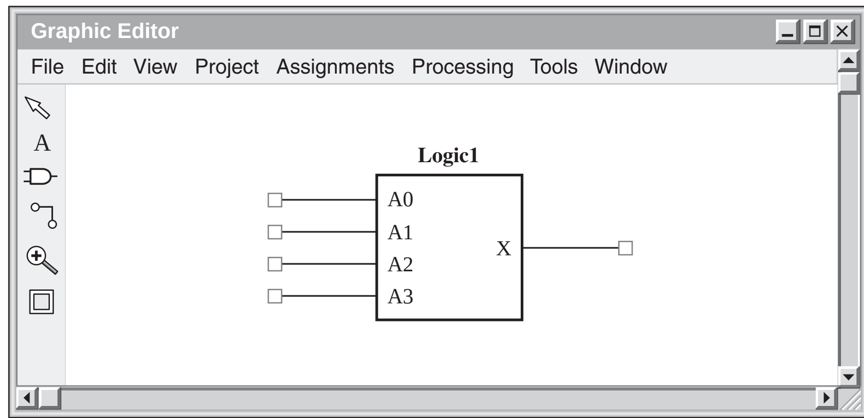
Vamos a utilizar el método jerárquico y a crear la lógica correspondiente a cada uno de los dos términos entre paréntesis incluidos en la ecuación anterior. Después, reduciremos cada circuito lógico a un único símbolo de bloque gráfico y a continuación, una vez completados ambos circuitos, colocaremos los dos bloques en la pantalla y conectaremos sus salidas a una puerta OR. Este proceso se ilustra en las cinco partes de la Figura 11.52. Podríamos haber introducido el circuito completo en la pantalla de una sola vez, pero este enfoque jerárquico resulta útil cuando el circuito lógico con el que estemos trabajando sea de gran tamaño y tenga que descomponerse en una serie de partes.

En la parte (e) de la Figura 11.52, el circuito lógico podría reducirse a otro símbolo de bloque y utilizarse en un diseño lógico todavía más grande; o bien podría guardarse y utilizarse en otros diseños, como se ilustra en la Figura 11.53.

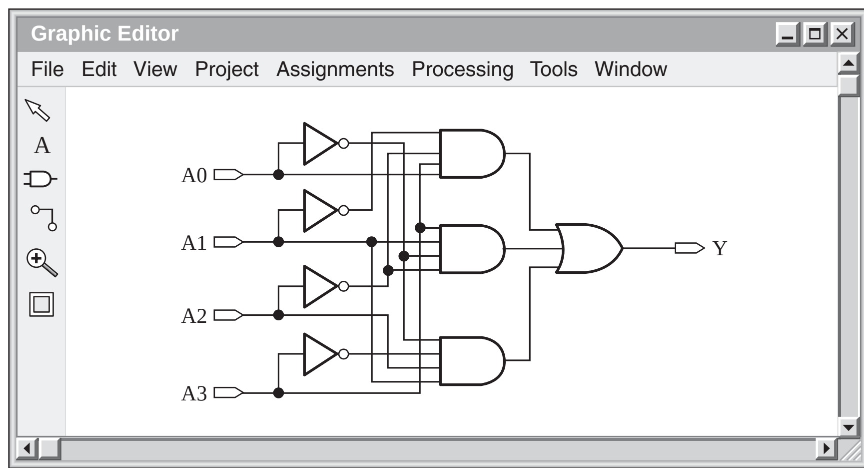


(a) Introducir la lógica AND-OR para el término compuesto por dos productos.

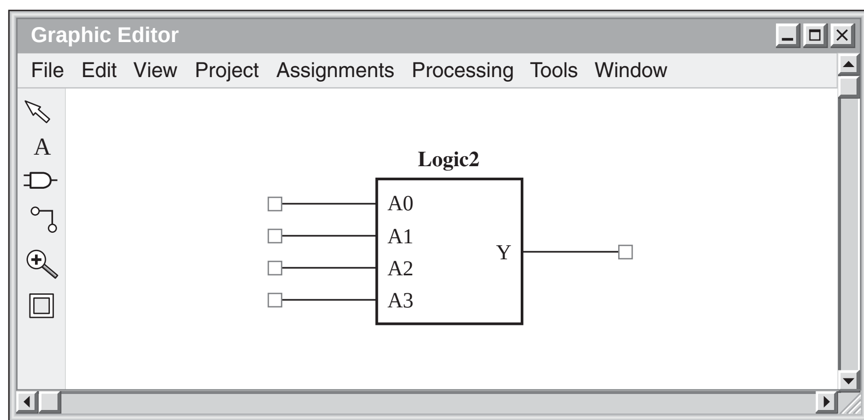
FIGURA 11.52 Ejemplo de creación de lógica en segmentos y de la posterior combinación de los segmentos. (*Continúa*)



(b) Reducir la lógica AND-OR a un símbolo de bloque definido como Logic1.

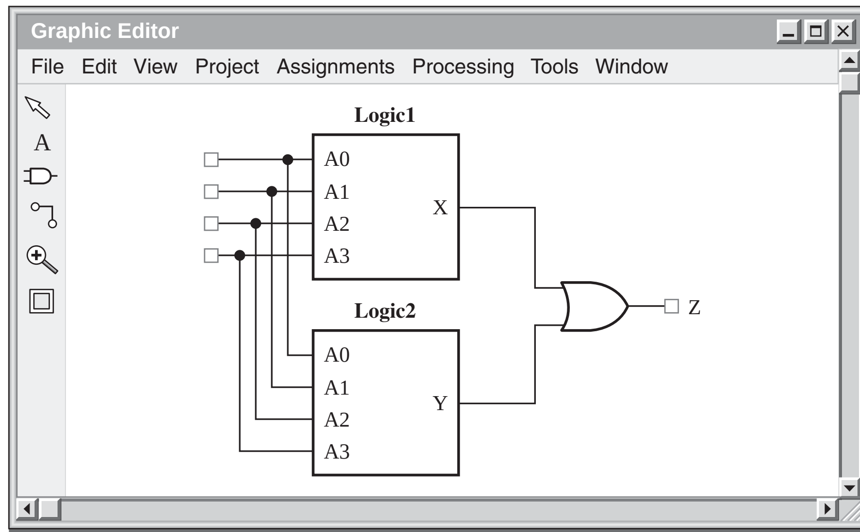


(c) Introducir la lógica AND-OR correspondiente al término de tres productos.



(d) Reducir a un símbolo de bloque definido como Logic2.

FIGURA 11.52 (*Continúa*)



(e) Combinar Logic1 y Logic2 mediante una puerta OR y conectar las entradas comunes.

FIGURA 11.52 (*Continuación*)

Después de haber introducido un circuito lógico en forma de esquemático, un programa de aplicación denominado ***compilador*** controla las diversas herramientas CAD que procesan el esquemático y generan una implementación para el dispositivo de destino.

Simulación funcional

El propósito de la simulación funcional dentro del flujo de diseño consiste en asegurarse de que el diseño introducido funciona como debería en términos de su operación lógica, antes de sintetizarlo en un diseño hardware. Básicamente, después de compilar un circuito lógico, se puede simular aplicando formas de onda de entrada y comprobando la salida para todas las posibles combinaciones de entrada, utilizándose en este proceso un editor de formas de onda.

El editor de formas de onda permite seleccionar los nodos (entradas y salidas) que se desea comprobar. Los nombres de las entradas y salidas seleccionadas aparecen en la pantalla del editor de formas de onda, junto con un símbolo o algún otro tipo de designación que identifica cada línea como una entrada o como una salida, tal como se muestra en la Figura 11.54. Inicialmente, las cuatro entradas tienen el valor predeterminado 0, y el tramado indica que el valor de salida es desconocido. Se puede seleccionar el intervalo de tiempo para la visualización.

A continuación, se crea cada forma de onda de entrada introduciendo un 1 o un 0 para cada intervalo de tiempo (intervalo entre las líneas de puntos de la Figura 11.55). Esto se suele llevar a cabo mediante un proceso que consiste en apuntar, hacer clic y seleccionar con el ratón, proceso que depende del software específico. Para este caso concreto, creamos las formas de onda de manera que se cubran las 16 posibles combinaciones de las cuatro entradas. La Figura 11.55 muestra las formas de onda de entrada (A0, A1, A2, A3) tal y como se las ha especificado.

Después de haber especificado las formas de onda de entrada, se abrirá generalmente una ventana de control de la simulación en la que podemos fijar los instantes inicial y final de la simulación y especificar los intervalos de tiempo que se quiere mostrar. Cuando arranca la simulación, se muestra la forma de onda de salida Z en el editor de formas de onda, como se ilustra en la Figura 11.56. Esto permite verificar que el diseño

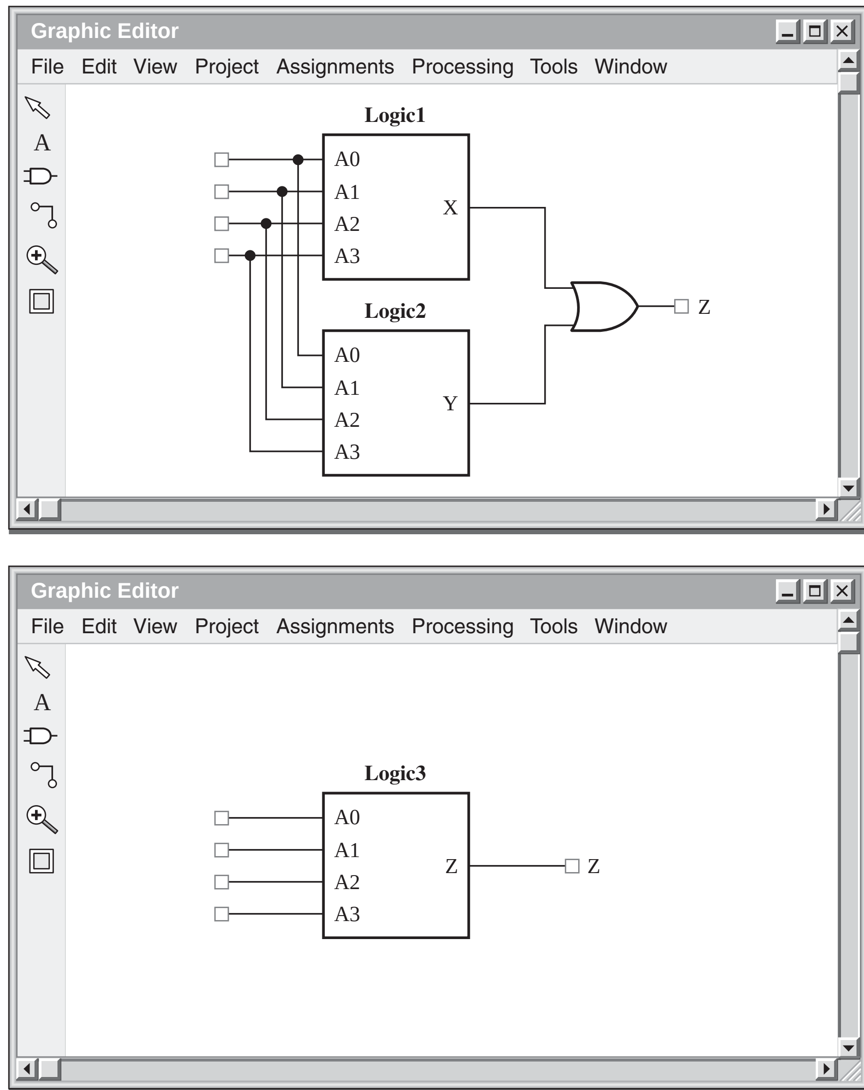


FIGURA 11.53 El circuito lógico con dos bloques lógicos y una puerta OR se reduce a otro bloque lógico, Logic3.

es correcto o que no está funcionando apropiadamente. En este caso, la forma de onda de salida es correcta para las formas de onda de entrada seleccionadas. Una forma de onda de salida incorrecta indicaría que existe algún fallo en la funcionalidad de la lógica, por lo que haría falta retroceder, comprobar el diseño original y volver a introducir el diseño revisado.

Síntesis

Una vez introducido el diseño y después de haberlo simulado funcionalmente para verificar que su operación es correcta, el compilador ejecuta automáticamente varias fases para preparar el diseño para su descarga en el dispositivo de destino. Durante la fase de síntesis del flujo de diseño, se optimiza el diseño con el objetivo de minimizar el número de puertas, sustituyendo algunos elementos lógicos por otros elementos que puedan

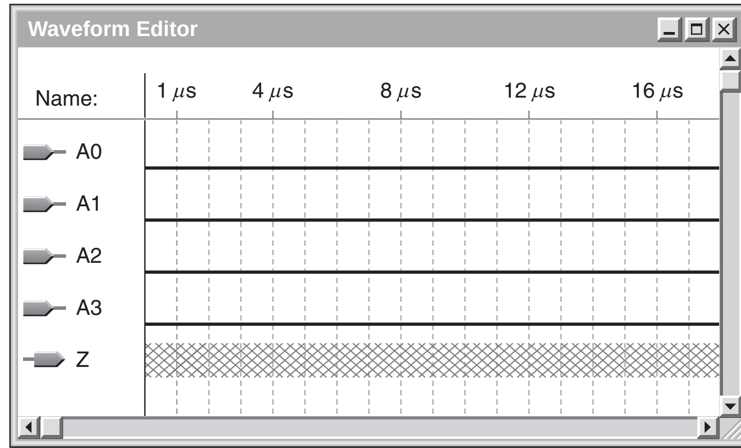


FIGURA 11.54 Pantalla genérica de un editor de formas de onda, donde se especifican los nombres de las entradas y salidas para el circuito introducido en la Figura 11.53.

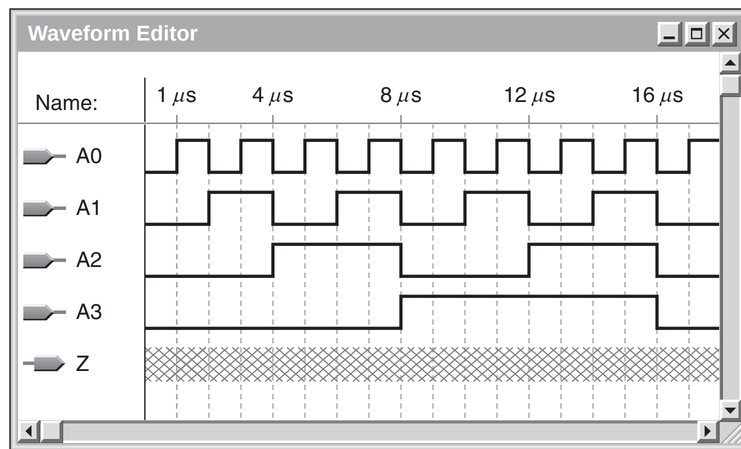


FIGURA 11.55 Formas de onda de entrada especificadas en la pantalla de editor de formas de onda.

realizar la función de manera más eficiente y eliminando cualquier lógica redundante. La salida final de la fase de síntesis es una lista de interconexiones que describe la versión optimizada del circuito lógico.

Lista de interconexiones. Una **lista de interconexiones** (*netlist*) es, básicamente, una lista de conectividad que describe los componentes y cómo están interconectados. Generalmente, una lista de interconexiones contiene referencias a las descripciones de los componentes o elementos utilizados. Cada vez que se utiliza en una lista de interconexiones un componente, como por ejemplo una puerta lógica, esa utilización se denomina *instancia.* Cada instancia tiene una definición que enumera las conexiones que pueden realizarse con ese tipo de componente, además de algunas propiedades básicas de dicho componente. Estos puntos de conexión se denominan *puertos* o *pines*. Usualmente, cada instancia tendrá un nombre distintivo, por ejemplo, si hay dos instancias de puerta AND, una será "and1" y la otra "and2". Dejando aparte sus nombres, dichas instancias pueden ser idénticas. Las interconexiones son los "hilos" que conectan entre sí los elementos del circuito. Las

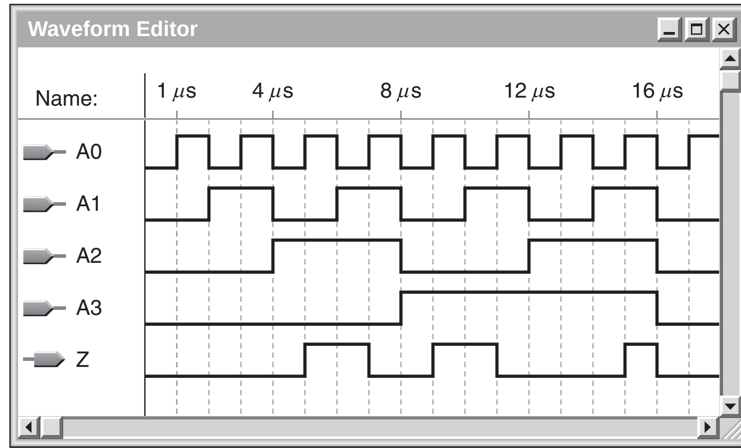


FIGURA 11.56 Después de ejecutar la simulación funcional, la forma de onda de salida debe indicar que la lógica está funcionando adecuadamente.

listas de interconexiones describen usualmente todas las instancias y sus atributos, y a continuación especifican cada una de las interconexiones, indicando los puertos de cada instancia a las que están conectadas.

El circuito lógico AND-OR introducido durante la fase de diseño y que se muestra en la Figura 11.57(a), podría dar como resultado el circuito optimizado mostrado en la parte (b) de la figura. En este ejemplo, el compilador ha eliminado las tres puertas OR y las ha sustituido por una única puerta OR de 5 entradas. Asimismo, se han eliminado dos inversores redundantes.

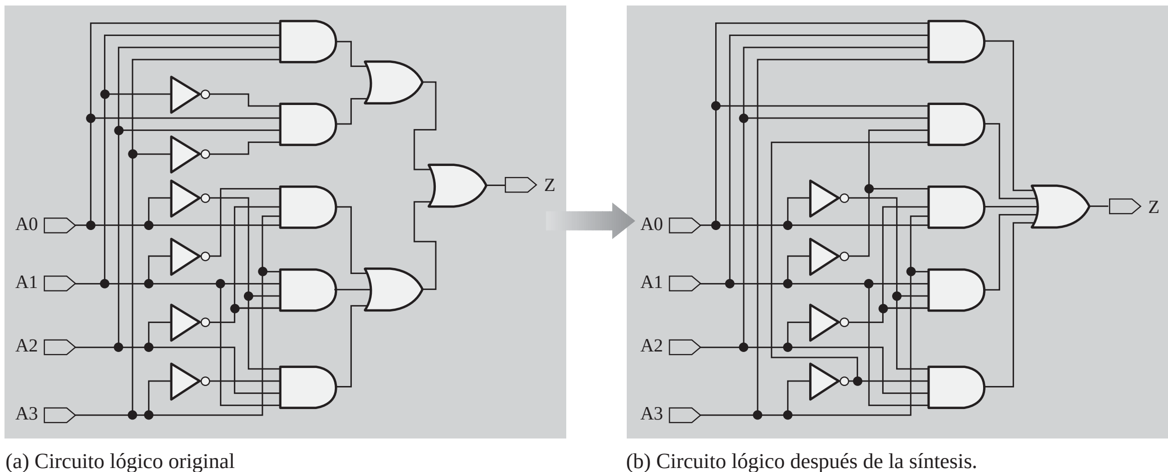


FIGURA 11.57 Ejemplo de optimización lógica durante la síntesis.

El software de síntesis genera una lista de interconexiones. Para ilustrar el concepto de lista de interconexiones genérica, la Figura 11.58(a) muestra las asignaciones de interconexiones, las asignaciones de instancias y las asignaciones de E/S. La lista de interconexiones mostrada en la Figura 11.58(b) no pretende ajustarse a ningún formato o sintaxis específicos de ninguna lista de interconexión, se trata de una lista de

interconexiones hipotética que simplemente indica el tipo de información necesaria para describir un circuito. Uno de los formatos utilizados en las listas de interconexiones es **EDIF** (*Electronic Design Interchange Format*, formato de intercambio para diseño electrónico).

Implementación (software)

Después de haber sintetizado el diseño, el compilador se encarga de implementarlo. Este proceso de implementación es, en esencia, un proceso de "mapeo" del diseño que busca encajar éste dentro del dispositivo de destino específico basándose en su arquitectura y de las configuraciones de pines. Este proceso se denomina *encaje* o *colocación y rutado*. Para poder llevar a cabo la fase de implementación del flujo de diseño, el software debe "conocer" los detalles relativos al dispositivo específico y disponer de información detallada acerca de los pines. Los datos completos de todos los dispositivos de destino potenciales suelen estar almacenados en la biblioteca software.

Simulación de temporización

Esta parte del flujo de diseño tiene lugar después de la implementación y antes de descargar el diseño en el dispositivo de destino. La simulación de temporización verifica que el circuito funciona a la frecuencia de diseño y que no existen retardos de propagación y otros problemas de temporización que vayan a afectar a la operación global. Puesto que ya se ha realizado una simulación funcional, el circuito debe funcionar de manera apropiada desde el punto de vista lógico. El software de desarrollo utiliza información acerca del dispositivo de destino específico, como por ejemplo los retardos de propagación de las puertas, con el fin de realizar una simulación de temporización del diseño. Para la simulación funcional no hacía falta la especificación del dispositivo de destino, para la simulación de temporización sí que es necesario elegir un dispositivo de destino específico. Puede utilizar el editor de formas de onda para visualizar el resultado de la simulación de

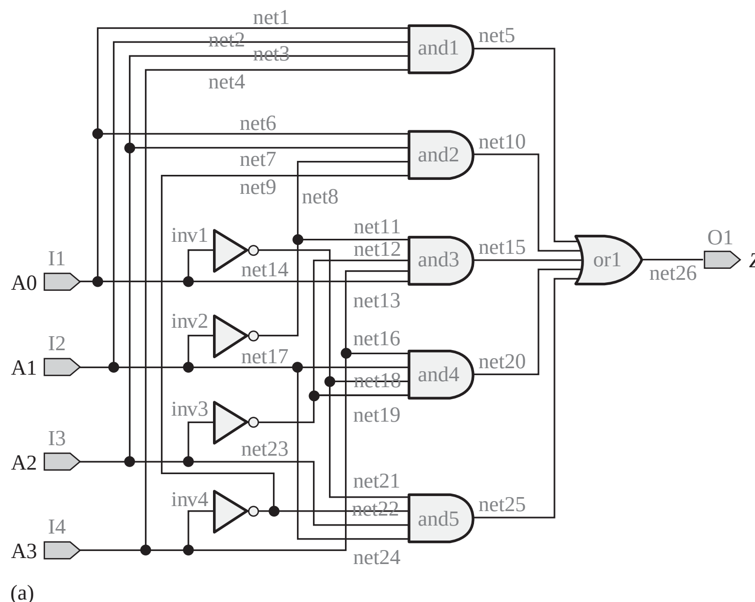


FIGURA 11.58 La fase de síntesis genera una lista de interconexiones para el circuito lógico optimizado. (*Continúa*)

```
Netlist (Logic3)
net<name>: instance<name>, <from>; <to>;
instances: and1, and2, and3, and4, and5, or1, inv1, inv2,
inv3, inv4;
Input/outputs: I1, I2, I3, I4, O1;
net1: and1, inport1; I1;
net2: and1, inport2; I2;
net3: and1, inport3; I3;
net4: and1, inport4; I4;
net5: and1, outport1; or1, inport1;
net6: and2, inport1; I1;
net7: and2, inport2; I3;
net8: and2, inport3; inv2, outport1;
net9: and2, inport4; inv4, outport1;
net10: and2, outport1; or1, inport2;
net11: and3, inport1; inv2, outport1;
net12: and3, inport2; inv3, outport1;
net13: and3, inport3; I4;
net14: and3, inport4; I1;
net15: and3, outport1; or1, inport3;
net16: and4, inport1; I4;
net17: and4, inport2; I2;
net18: and4, inport3; inv1, outport1;
net19: and4, inport4; inv3, outport1;
net20: and4, outport1; or1, inport4;
net21: and5, inport1; inv1, outport1;
net22: and5, inport2; inv4, outport1;
net23: and5, inport3; I3;
net24: and5, inport4; I2;
net25: and5, outport1; or1, inport5;
net26: or1, outport1; O1;
end
```

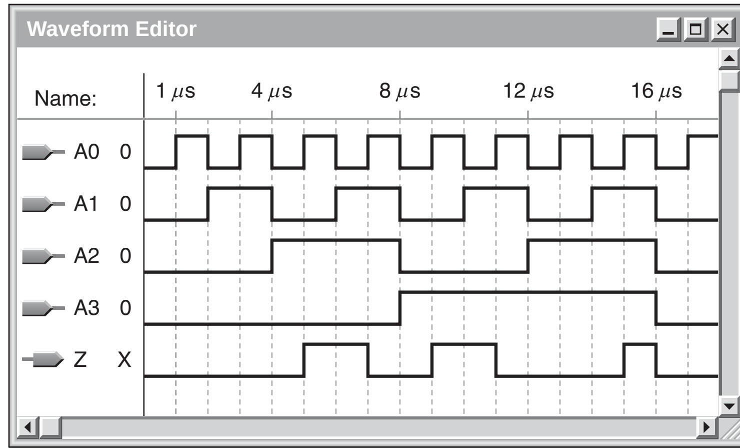
(b)

FIGURA 11.58 (*Continuación*)

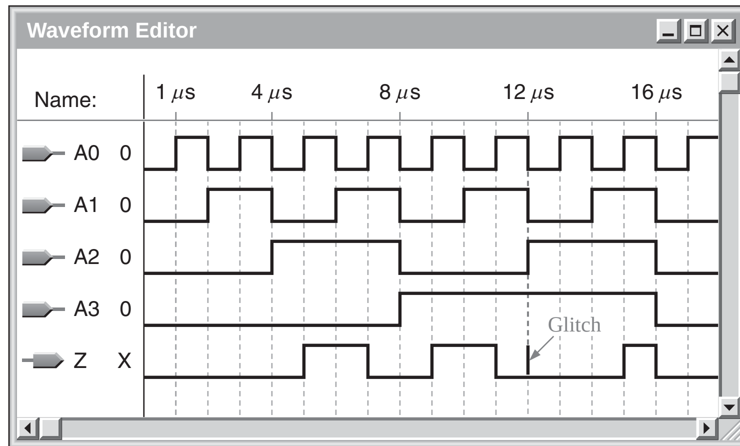
temporización, al igual que hacíamos en la simulación funcional y tal como se ilustra en la Figura 11.59. Si no existe ningún problema relativo a la temporización, como se muestra en la parte (a) de la figura, el diseño estará listo para ser descargado. Sin embargo, suponga que la simulación de temporización revela la presencia de un "*glitch*" debido al retardo de propagación, como se muestra en la Figura 11.59(b). Un *glitch* es un pico de muy corta duración en la forma de onda. En este caso, será necesario analizar cuidadosamente el diseño para localizar la causa y luego volver a introducir el diseño modificado y repetir el proceso de flujo de diseño. Recuerde que todavía no hemos implementado en hardware el diseño.

Programación del dispositivo (descarga)

Después de haber verificado mediante las simulaciones funcional y de temporización que el diseño está funcionando correctamente, podemos iniciar la secuencia de descarga. Para ello se genera un **flujo de bits**, que representa el diseño final, y ese flujo de bits se envía al dispositivo de destino con el fin de configurarlo automáticamente. Después de completar este proceso, el diseño estará implementado en el hardware y podrá comprobarse dentro del circuito. La Figura 11.60 ilustra el concepto básico de ***descarga***.



(a) Resultado correcto



(b) Problema de temporización

FIGURA 11.59 Ejemplos hipotéticos de resultados de la simulación de temporización.

Analogía urbanística

Una forma de entender los procesos de implementación y de descarga consiste en utilizar una analogía de desarrollo urbanístico. El desarrollador comienza con un determinado terreno, lo analiza y lo divide en una serie de parcelas (lo que sería análogo a tener un dispositivo no programado). A continuación, se concibe un plano del desarrollo urbanístico y se disponen todas las parcelas, edificios, calles y servicios (lo que sería análogo a la introducción del diseño). Después, es necesario verificar que el número de edificios y que la infraestructura encajan en el terreno disponible y cumplen con toda la normativa local. El plano de la actuación urbanística también muestra la localización de cada edificio y la disposición de cada calle y acera (lo que sería análogo a la síntesis y a la implementación). Hasta que no ha tenido lugar este proceso de disposición sobre el mapa de terreno, el promotor no puede comenzar a construir físicamente los edificios, calles y servicios (lo que sería análogo a la descarga). Esta analogía urbanística se ilustra en la Figura 11.61(a). El proceso de implementación de un diseño lógico en un dispositivo programable se muestra en la parte (b) de la figura, donde la fase

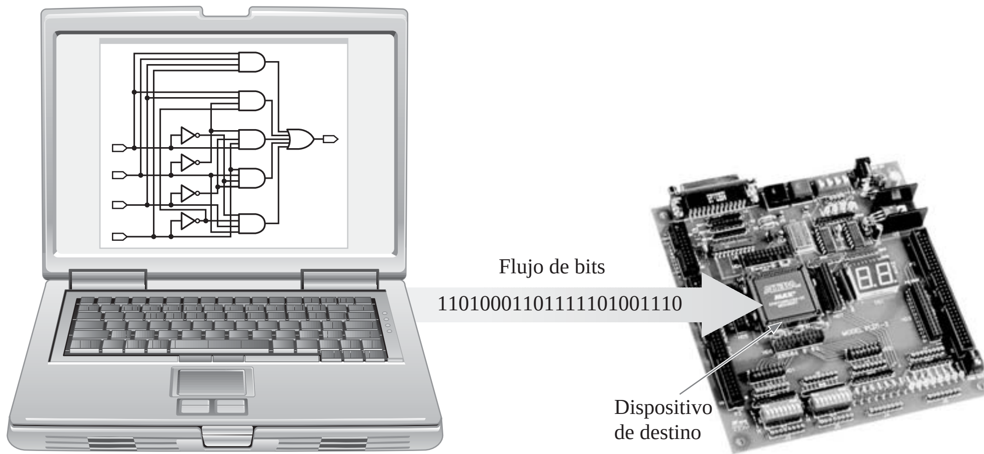


FIGURA 11.60 Descarga de un diseño en el dispositivo de destino.

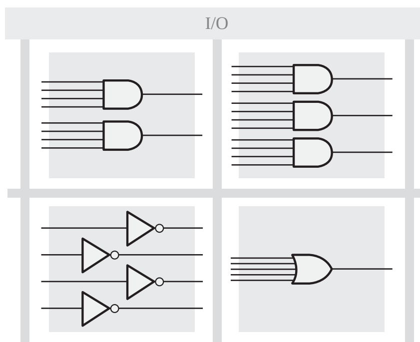


Es necesario completar la actuación urbanística antes de implementar físicamente dicha actuación. Esto es análogo a la fase de implementación del flujo de diseño en el caso de la lógica programable.

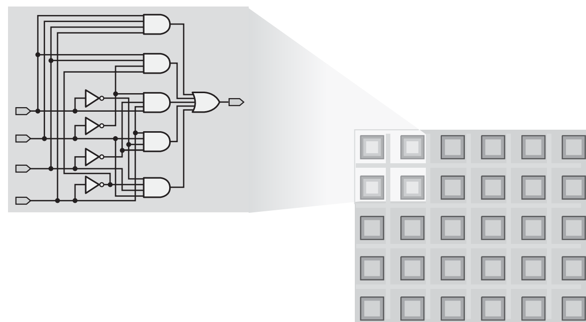


Las estructuras físicas se colocan sólo después de que el plano de la actuación urbanística haya permitido verificar si todos los elementos van a poder encajar dentro del terreno disponible. Esto es análogo a descargar el diseño en el dispositivo de destino.

(a)



Aplicación del diseño software sobre el "mapa" del dispositivo de destino. Esto es análogo al plano urbanístico.



Descarga del diseño en el dispositivo de destino (análogo a la colocación de las estructuras físicas).

(b)

FIGURA 11.61 Analogía de una promoción inmobiliaria para ilustrar la implementación (plano urbanístico) y descarga (construcción) de un diseño lógico en un dispositivo de destino.

de mapeo es análoga al desarrollo del plan de actuación urbanística y la fase de descarga de la colocación de las estructuras físicas sobre el terreno.

REVISIÓN DE LA SECCIÓN 11.8

1. Enumerar las fases del flujo de diseño para la lógica programable.
2. Enumerar los elementos esenciales para programar un CPLD o una FPGA.
3. ¿Cuál es el propósito de una lista de interconexiones?
4. ¿Qué fase tiene lugar antes dentro del flujo de diseño: la simulación funcional o la simulación de temporización?

11.9 LÓGICA DE EXPLORACIÓN DE CONTORNO (BOUNDARY SCAN LOGIC)

La técnica de exploración de contorno se utiliza tanto para la prueba como para la programación de la lógica interna de un dispositivo programable. El estándar JTAG para la lógica de la exploración de contornos se especifica en el documento IEEE Std. 1149.1. La mayoría de los dispositivos lógicos programables son compatibles con JTAG. En esta sección, vamos a presentar la arquitectura básica de un dispositivo JTAG IEEE Std. 1149.1 y la analizaremos en función de los detalles de su registro de exploración de contorno y de la estructura de la lógica de control.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Describir los elementos requeridos en un dispositivo compatible con JTAG. ■ Indicar las entradas y salidas JTAG obligatorias. ■ Indicar el propósito del registro de exploración de contorno. ■ Indicar el propósito del registro de instrucciones. ■ Indicar para qué se utiliza el registro de puenteo.

Registros IEEE Std. 1149.1

Todos los registros de lógica de programable que son compatibles con el estándar IEEE Std. 1149.1 requieren los elementos que se muestran en el diagrama simplificado de la Figura 11.62. Se trata del registro de exploración de contorno, el registro puenteo, el registro de instrucción y la lógica TAP (*Test Access Port*, puerto de acceso de pruebas). Un quinto registro, el registro de identificación, es opcional y no se muestra en la figura.

Registro de exploración de contorno (BS, boundary Scan). Las celdas de exploración de contorno (BSC, *Boundary Scan Cell*) interconectadas forman el registro de exploración de contorno. La entrada serie al registro es la línea TDI (*Test Data In*, entrada de datos de prueba) y la salida serie es TDO (*Test Data Out*, datos de salida de prueba). Los datos correspondientes a la lógica interna y a los pines de entrada y de salida del dispositivo también se pueden desplazar de forma paralela para introducirlos en el registro BS. El registro BS se emplea para probar las conexiones entre los dispositivos PLD y para probar la lógica interna que haya sido programada en el dispositivo.

Registro de puenteo (BP, Bypass). Este registro de datos obligatorio (que normalmente es un único flip-flop) optimiza el método de desplazamiento, acortando el camino entre la línea TDI y la línea TDO para aquellos casos en que no se utilice el registro BS ni ningún otro registro de datos.

Registro de instrucción. Este registro obligatorio almacena las instrucciones para la ejecución de diversas operaciones de exploración de contorno.

Registro de identificación (ID, Identification). El registro de identificación es un registro de datos opcional que el estándar IEEE Std. 1149.1 no considera obligatorio. Sin embargo, se emplea en algunas arquitecturas de exploración de contorno para almacenar un código que identifica el dispositivo programable concreto que se esté usando.

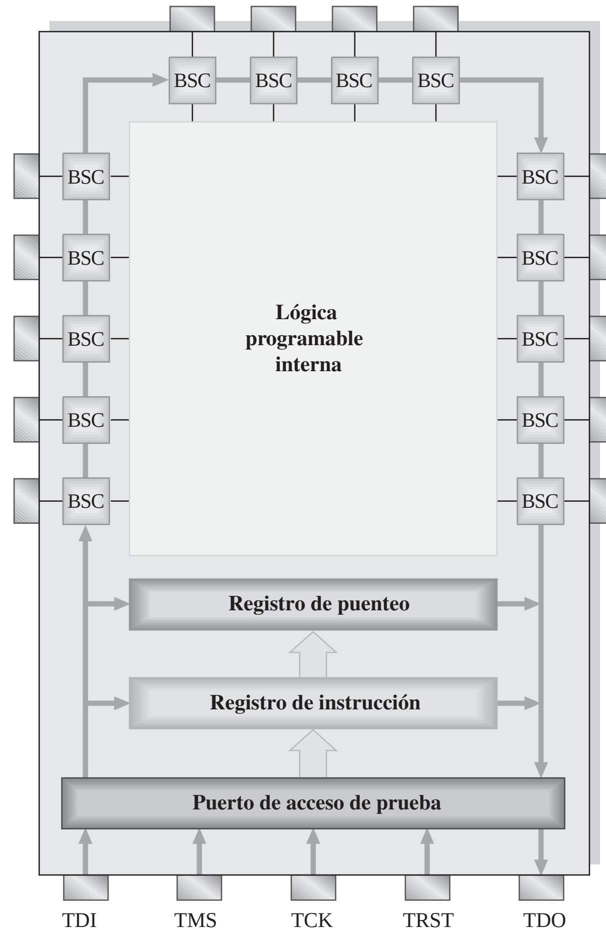


FIGURA 11.62 Diagrama muy simplificado de un dispositivo de lógica programable (CPLD o FPGA) compatible con JTAG (IEEE Std. 1149.1). Las celdas BSC forman el registro de exploración de contorno. Sólo se muestra en la figura un pequeño número de celdas BSC, como ilustración.

Instrucciones de exploración de contorno en el estándar IEEE Std. 1149.1

Se utilizan varias instrucciones estándar para controlar la lógica de exploración de contorno. Además de las que a continuación se indican, hay disponibles otras instrucciones opcionales.

- *BYPASS* Esta instrucción conmuta el registro BP para introducirlo en la ruta TDI/TDO.
- *EXTEST* Esta instrucción conmuta el registro BS para introducirlo en la ruta TDI/TDO y permite realizar pruebas de pines externo y pruebas de interconexión entre la salida de un dispositivo lógico programable y la entrada de otro.
- *INTEST* Esta instrucción conmuta el registro BS para introducirlo en la ruta TDI/TDO y permite probar la lógica interna programada.
- *SAMPLE/PRELOAD* Esta instrucción se utiliza para muestrear los datos en los pines de entrada del dispositivo y aplicar esos datos a la lógica interna. También se emplea para aplicar datos (precarga) procedentes de la lógica interna a los pines de salida del dispositivo.

- *IDCODE* Esta instrucción conmuta el registro de identificación opcional para introducirlo dentro de la ruta TDI/TDO, de modo que el código ID pueda desplazarse y suministrarse como salida a través de la línea TDO.

Puerto TAP del estándar IEEE Std. 1149.1

El puerto de acceso de prueba (TAP) consta de una lógica de control, cuatro entradas y salidas obligatorias y una entrada opcional que se denomina Test Reset (TRST).

- *TDI (Test Data In)* La línea TDI permite introducir mediante desplazamiento serie los datos de prueba y de programación, así como las instrucciones dentro de la lógica de exploración de contorno.
- *TDO (Test Data Out)* La línea TDO permite extraer de la lógica de exploración de contorno los datos de prueba y de programación, así como las instrucciones mediante un desplazamiento serie.
- *TMS (Test Mode Select)* La línea TMS permite conmutar entre los distintos estados del controlador TAP.
- *TCK (Test Clock)* La línea TCK proporciona la temporización para el controlador TAP, a partir de la cual se generan las señales de control para los registros de datos y para el registro de instrucción.

En la Figura 11.63 se muestra un diagrama de bloques de la lógica de exploración de contorno. A través de la línea TDI se introducen por desplazamiento tanto las instrucciones como los datos. El controlador TAP dirige las instrucciones al registro de instrucción y los datos al registro de datos apropiado. Una instrucción

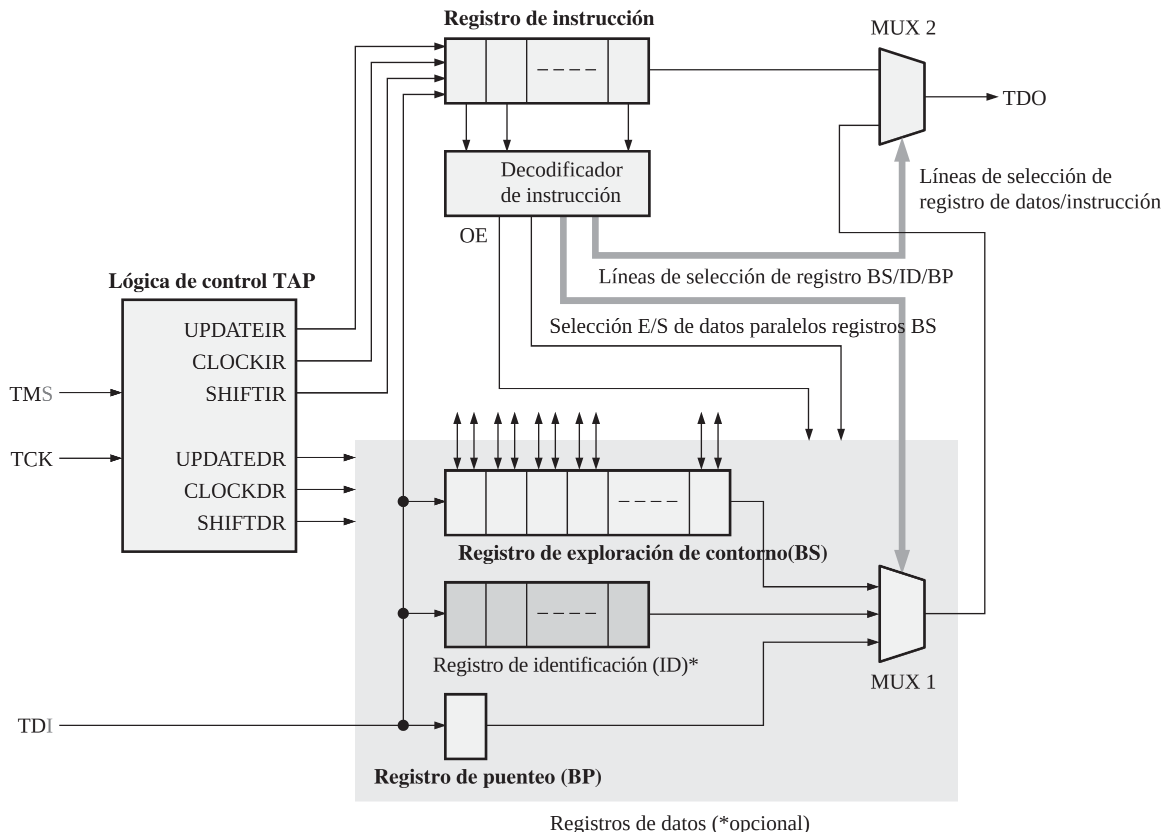


FIGURA 11.63 Diagrama lógico de la exploración de contorno.

decodificada mediante el decodificador de instrucciones selecciona el registro de datos al que hay que acceder a través de MUX 1 y también selecciona, mediante MUX 2, si lo que hay que desplazar hacia afuera a través de la línea TDO es una instrucción o un dato. Asimismo, la instrucción decodificada permite configurar el registro de exploración de contorno en uno de cinco posibles modos básicos. A continuación se describen la celda de operación de contorno y sus modos de operación.

La celda BSC

El registro de exploración de contorno está formado por celdas BSC (*Boundary Scan Cell*). En la Figura 11.64 se muestra un diagrama de bloques de una celda BSC básica. Como se indica, los datos pueden introducirse y extraerse de la celda BSC mediante desplazamiento en serie. Asimismo, los datos pueden introducirse en la celda BSC a partir de la lógica programable interna, a partir de un pin de entrada del dispositivo o a partir de la BSC anterior. Además, los datos pueden extraerse de la celda BSC para dirigirlos a la lógica programable interna, a un pin de salida del dispositivo o a la siguiente BSC.

En la Figura 11.65 se muestra la arquitectura de una celda genérica de exploración de contorno. La celda está compuesta por dos circuitos lógicos idénticos, cada uno de los cuales contiene dos flip-flops y dos multiplexores. Esencialmente, uno de los circuitos permite introducir los datos desde la lógica programable interna o extraerlos a un pin de salida del dispositivo, mientras que el otro circuito permite introducir los datos desde un pin de entrada del circuito o permite extraerlos hacia la lógica programable interna.

Hay cinco modos en los que la celda BSC puede operar, en términos del flujo de datos. El primer modo BSC permite que los datos fluyan en serie desde la celda BSC anterior hasta la siguiente, como se ilustra en la Figura 11.66. Un 1 en la entrada SHIFT selecciona la línea SDI. Los datos de la línea SDI se enclavan en el registro de captura A en el flanco positivo de la señal de reloj (CLOCK). A continuación los datos se enclavan en el registro de captura B con el flanco negativo de la señal de reloj y aparecen en la línea SDO. Esto es equivalente a desplazar en serie los datos a través de un registro de exploración de contorno.

El segundo modo BSC permite que los datos fluyan desde la lógica programable interna a un pin de salida del dispositivo, como se ilustra en la Figura 11.67. El valor 0 en la línea de control PDI/O (*Parallel Data* I/O, E/S de datos en paralelo) selecciona los datos de la lógica programable interna. El valor 1 en la línea OE (*Output Enable,* habilitación de salida) activa el buffer de salida.

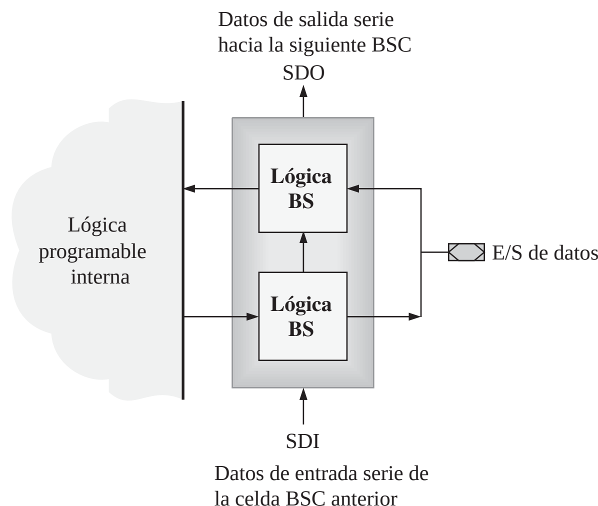


FIGURA 11.64 Un celda SC básica bidireccional.

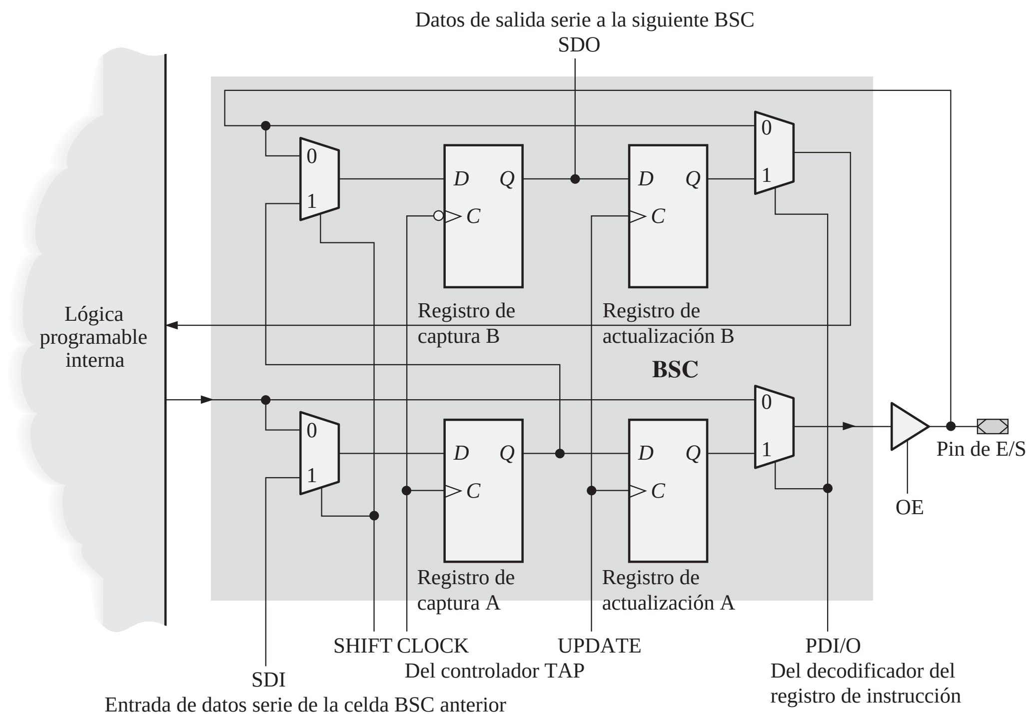


FIGURA 11.65 Arquitectura representativa de una típica celda de exploración de contorno.

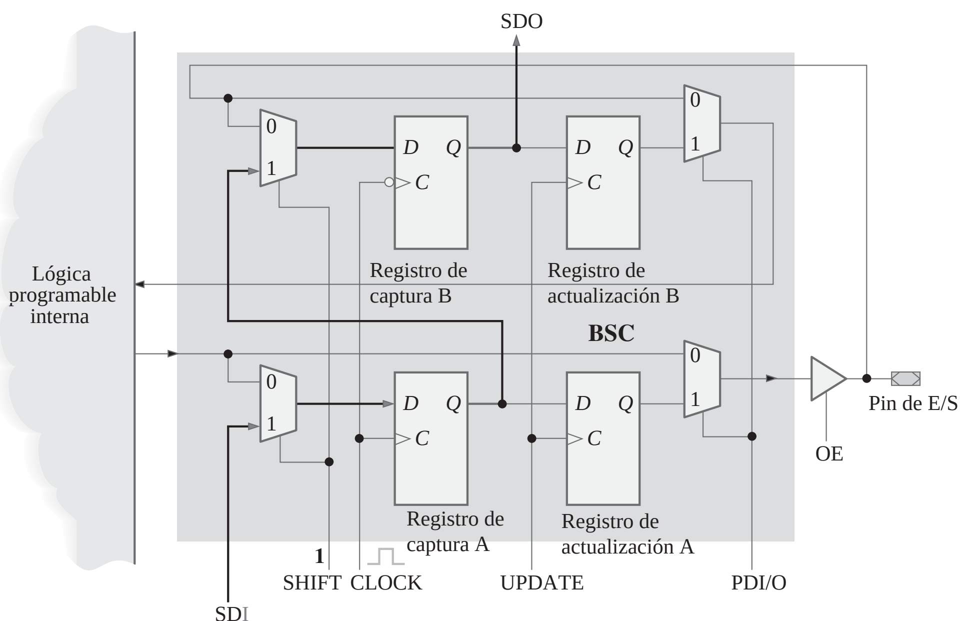


FIGURA 11.66 Camino de los datos para introducir en serie en una celda BSC los datos procedentes de la anterior. Hay un 1 en la entrada SHIFT y se aplica un pulso a la línea CLOCK. Las líneas en negro indican el flujo de los datos.

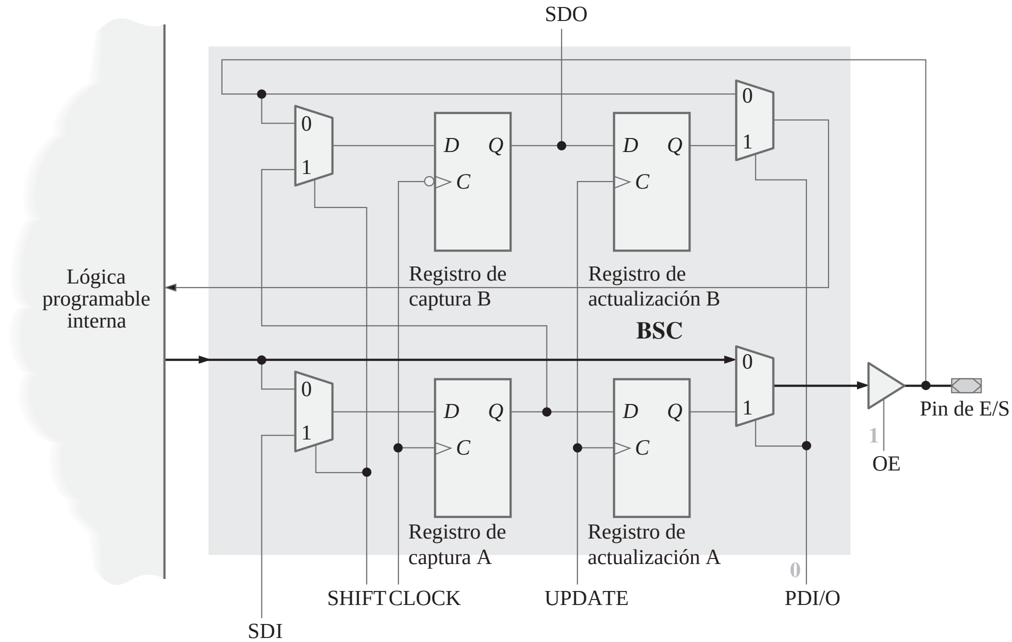


FIGURA 11.67 Camino de los datos para transferir datos desde la lógica programable interna hasta un pin de salida del dispositivo. Hay un 0 en la línea PDI/O y un 1 en la línea OE.

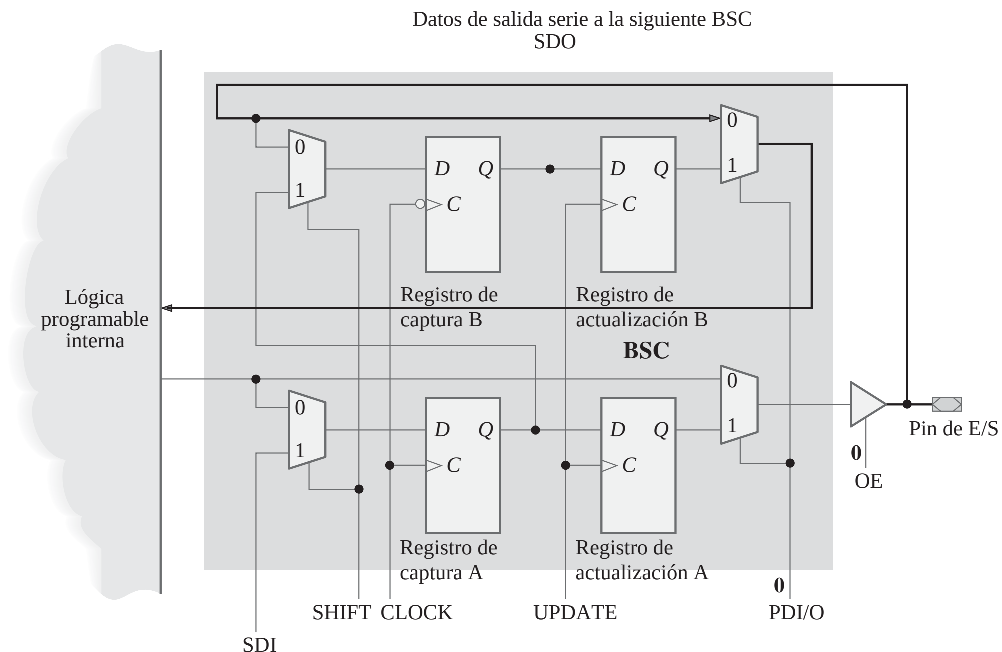


FIGURA 11.68 Camino de los datos para transferir datos desde un pin de entrada del dispositivo a la lógica interna programable. Hay un 0 en la línea PDI/O y un 0 en la línea OE.

El tercer modo BSC permite que los datos fluyan directamente desde un pin de entrada del dispositivo hasta la lógica programable interna, como se ilustra en la Figura 11.68. El valor 0 en la línea de control PDI/O selecciona los datos procedentes del pin de entrada. El valor 0 en la línea OE desactiva el *buffer* de salida.

El cuarto modo BSC permite que los datos fluyan desde la línea SDI hacia la lógica programable interna, como se ilustra en la Figura 11.69. Un valor 1 en la entrada SHIFT selecciona la línea SDI. Los datos de la línea SDI se enclavan en el registro de captura A con el flanco positivo de la señal CLOCK. A continuación, se enclavan en el registro de captura B con el flanco negativo de la señal CLOCK y aparecen en la SDO. Un pulso en la línea UPDATE enclava los datos en el registro de actualización B. Un 1 en la línea PDI/O selecciona la salida del registro de actualización B y la aplica a la lógica programable interna. Los datos aparecen también en la línea SDO.

El quinto modo BSC permite que los datos fluyan desde la línea SDI hasta un pin de salida del dispositivo y hacia la salida SDO, como se ilustra en la Figura 11.70. Un valor 1 en la entrada SHIFT selecciona la línea SDI. Los datos de la línea SDI se enclavan en el registro de captura A con el flanco positivo de la señal CLOCK. A continuación, se enclavan en el registro de captura B con el flanco negativo de la señal CLOCK y aparecen en la SDO. Un pulso en la línea UPDATE enclava los datos en el registro de actualización A. Con un valor 1 en la línea OE, un valor 1 en la PDI/O seleccionará la salida del registro de actualización A y la aplicará al pin de salida del dispositivo.

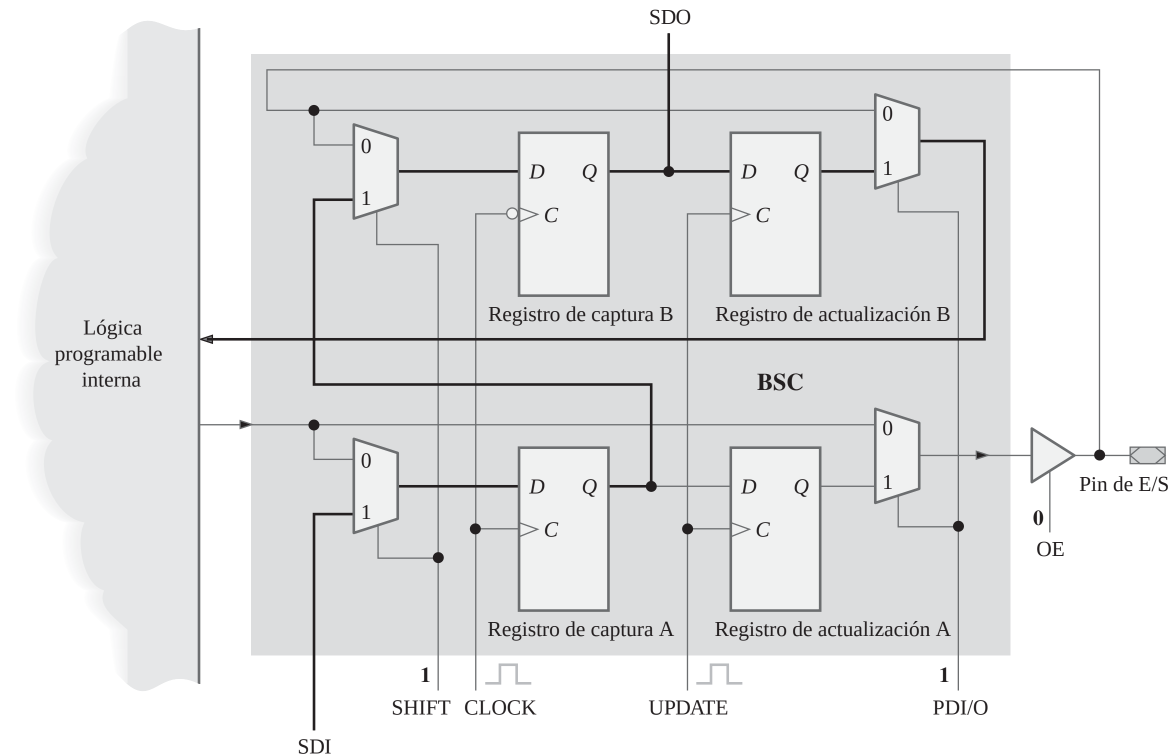


FIGURA 11.69 Camino de los datos para transferir datos desde la línea SDI hacia la lógica interna programable y hacia la línea SDO. Hay un 1 en la línea SHIFT, un 1 en la línea PDI/O y un 0 en la línea OE. Se aplica un pulso a la línea CLOCK seguido de un pulso en la señal UPDATE.

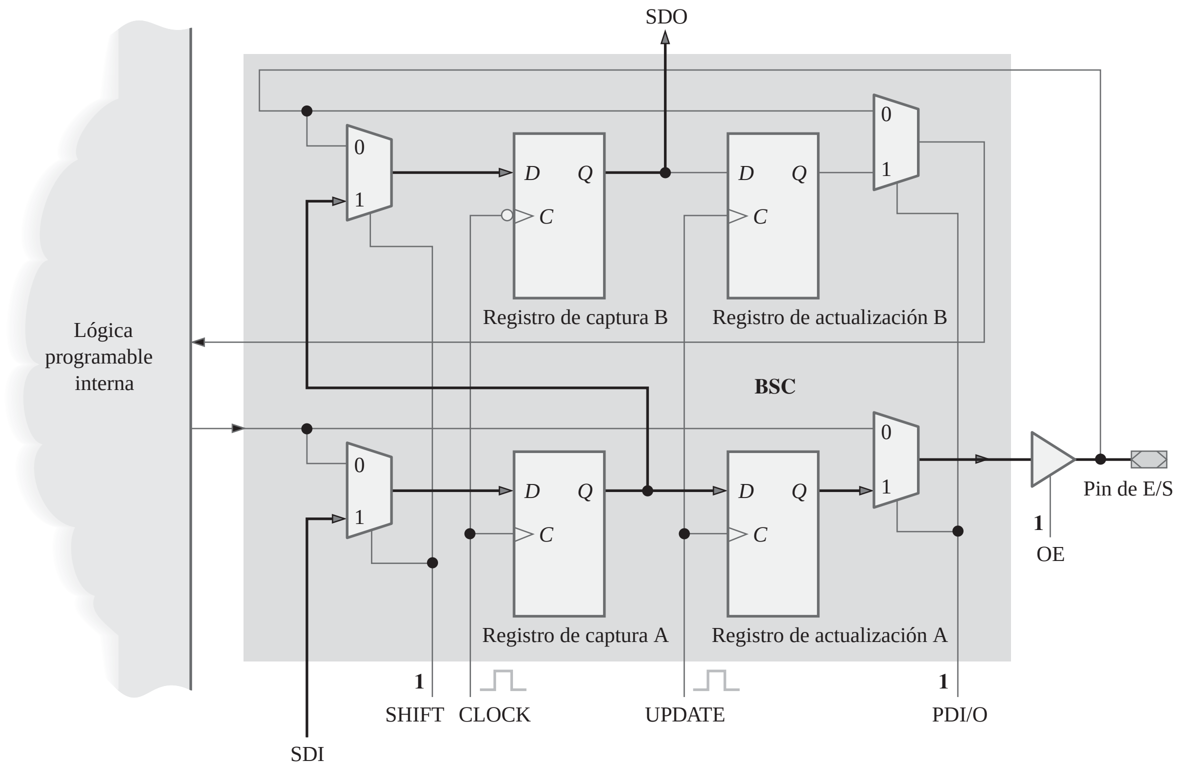


FIGURA 11.70 Camino de los datos para transferir datos desde la línea SDI hacia un pin de salida del dispositivo y hacia la línea SDO. Hay un 1 en la línea SHIFT, un 1 en la línea PDI/O y un 1 en la línea OE. Se aplica un pulso a la línea CLOCK seguido de un pulso en la señal UPDATE.

Prueba de exploración de contorno de múltiples dispositivos

Pueden aplicarse pruebas de exploración de contorno a tarjetas en las que se monten múltiples dispositivos JTAG (IEEE Std. 1149.1) para comprobar tanto las interconexiones como la lógica interna. Este concepto puede ilustrarse siguiendo el camino de los datos que se muestra en la Figura 11.71.

Cada bit se introduce por desplazamiento en la línea TDI del dispositivo 1 y a través del registro BS del dispositivo 1, hasta llegar a una celda en la que la conexión que hay que comprobar va al dispositivo 2. El bit se extrae por desplazamiento a través del pin de salida del dispositivo y pasa por la interconexión, llegando al pin de entrada del dispositivo 2. El bit continúa a través del registro BS del dispositivo 2, hasta alcanzar un pin de salida y pasar por la interconexión que lleva al pin de entrada del dispositivo 3. A continuación se desplaza a través del registro BS del dispositivo 3 hasta alcanzar la línea TDO. Si el bit que sale por la línea TDO es igual que el que entró por la línea TDI, las celdas de exploración de contorno a través de las que se ha desplazado y las interconexiones entre el dispositivo 1 y el dispositivo 2 y el dispositivo 2 y el dispositivo 3 son correctas.

REVISIÓN DE LA SECCIÓN 11.9

1. Enumerar las entradas y salidas de exploración de contorno requeridas por el estándar IEEE Std. 1149.1
2. ¿Qué es el puerto TAP?
3. Indicar los registros obligatorios en la lógica de exploración de contorno.

REVISIÓN DE LA SECCIÓN 11.9

4. Describir los cinco modos en los que puede operar una celda de exploración de contorno en términos del flujo de datos.

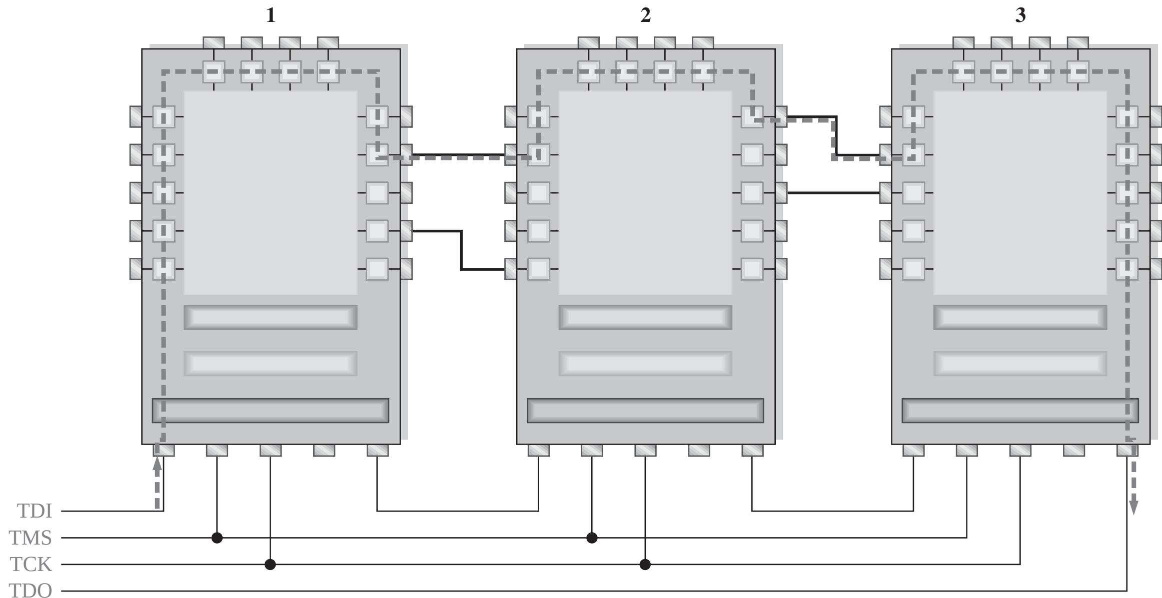


FIGURA 11.71 Concepto básico de pruebas de exploración de contorno para múltiples dispositivos e interconexiones. La ruta de prueba se muestra con línea de puntos.

11.10 LOCALIZACIÓN DE AVERÍAS

Hay dos formas básicas de probar un dispositivo que ha sido programado con un diseño lógico: la forma tradicional y la forma automatizada. Según el método tradicional, pueden emplearse instrumentos comunes de prueba en laboratorio para comprobar el funcionamiento del dispositivo. Con el método automático pueden utilizarse tres métodos fundamentales de prueba: cama de pinchos, sonda volante y exploración de contornos.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Describir el método de pruebas tradicional. ■ Describir las pruebas con camas de pinchos y con sonda volante y explicar sus limitaciones. ■ Explicar el estándar JTAG. ■ Describir el concepto básico de exploración de contorno. ■ Explicar los modos de prueba mediante exploración de contorno e indicar brevemente el lenguaje BSDL.

Después de implementar un diseño lógico en hardware, se puede probar el dispositivo en una tarjeta de circuito impreso. Para diseños relativamente simples, puede probarse el dispositivo utilizando instrumentos normales de prueba en laboratorio, como un osciloscopio o un analizador lógico, un generador de señales y una fuente de alimentación continua. Pueden aplicarse señales de entrada a los pines de entrada de la tarjeta y comprobar si en los pines de salida aparecen las formas de onda correctas. Esta técnica tradicional, ilustrada en la Figura 11.72, resulta práctica para tarjetas de evaluación que no vayan a fabricarse en serie y para las pruebas de pre-producción de los prototipos de circuitos.

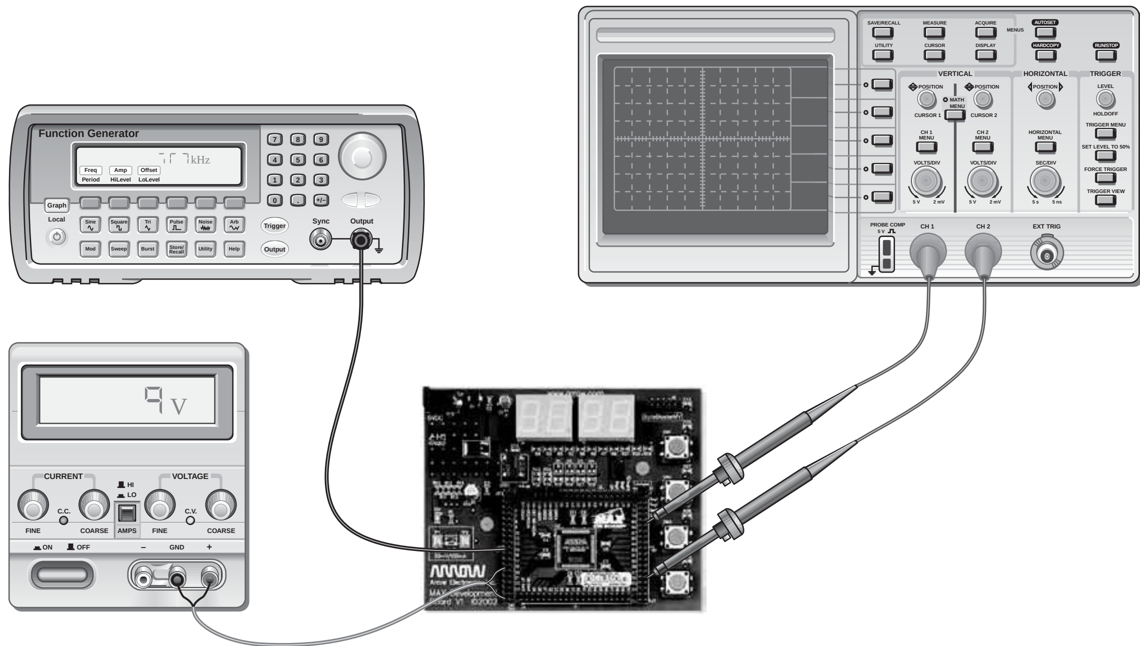


FIGURA 11.72 Pruebas tradicionales utilizando instrumentos de laboratorio.

Pruebas mediante cama de pinchos

La prueba de las tarjetas de circuito impreso en niveles de producción debe llevarse a cabo automáticamente. El método de la ***cama de pinchos*** fue una de las primeras técnicas de prueba automática utilizadas. El concepto se ilustra en la Figura 11.73, donde se ve cómo la tarjeta del circuito impreso se coloca sobre un

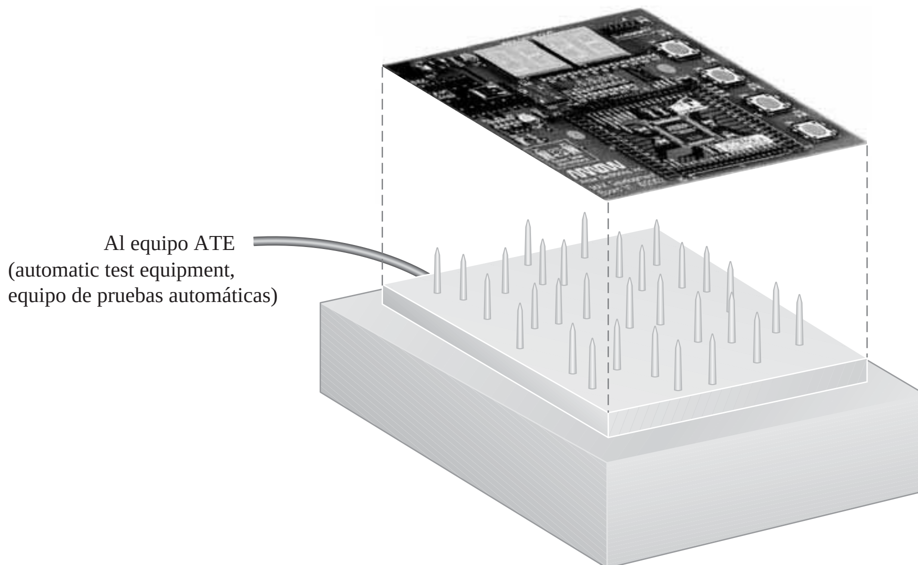


FIGURA 11.73 Concepto en que se basa el método de cama de pinchos para probar tarjetas de circuito.

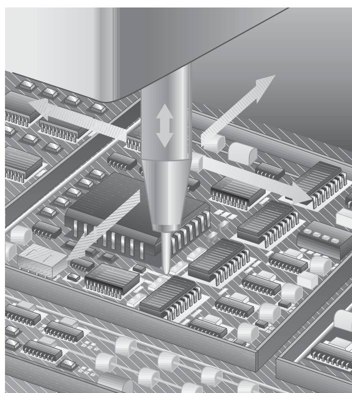
aparato que dispone de una matriz de pequeñas sondas de prueba similares a pinchos, que hacen contacto con los nodos de prueba de la tarjeta. Los "pinchos" se disponen según una matriz que se corresponde con el patrón de nodos de prueba de la tarjeta. Con este método, los puntos de prueba pueden comprobarse simultáneamente, utilizando equipos especiales de prueba automática. Básicamente, el propósito de las pruebas de producción automatizadas consiste en localizar todos los fallos de fabricación, como por ejemplo la existencia de pines en circuito abierto o cortocircuito o la presencia de componentes erróneos, componentes que falten o componentes desalineados. Este proceso automatizado no trata principalmente de comprobar la funcionalidad de la lógica. Se supone que se ha probado la funcionalidad de cada componente antes de montarlo en la tarjeta de circuito y que los únicos fallos posibles son aquellos que se hayan producido durante la fabricación.

A medida que los CI se fueron haciendo más pequeños y complejos, se aceleró la tendencia hacia la utilización de tecnología de montaje superficial y las tarjetas de circuito impreso cambiaron, pasando de ser de doble capa a ser multicapa. La mayor densidad y complejidad de las tarjetas de circuito y de los dispositivos, que ahora tenían un gran número de pines muy próximos entre sí, hicieron que fuera más difícil acceder a los puntos de prueba de la tarjeta utilizando la técnica de cama de pinchos.

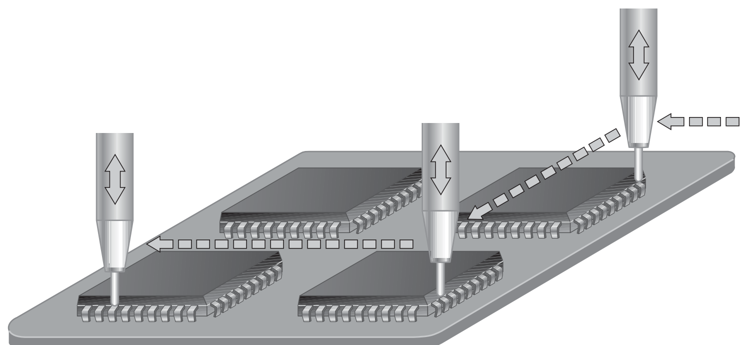
Prueba mediante sonda volante

Otro método para probar tarjetas de circuito impreso es el que se denomina método de ***sonda volante***. En la Figura 11.74 se muestra una sonda volante típica y su forma de operación básica. Se coloca una sonda de prueba sobre la tarjeta de circuito que haya que comprobar. La sonda puede moverse automáticamente a lo largo de los tres ejes de la tarjeta (eje x, eje y y eje z) con el fin de entrar en contacto con cualquier punto de prueba especificado. El movimiento de la sonda se controla mediante un software que utiliza la disposición física de la tarjeta para determinar las coordenadas. Muchos instrumentos de prueba basados en sonda volante disponen de múltiples sondas para probar una tarjeta.

El método de prueba basado en sonda volante resuelve algunas de las limitaciones de la cama de pinchos. En primer lugar, el método de la cama de pinchos requiere un aparato distinto para cada tipo de tarjeta de circuito, mientras que el método basado en sonda volante no requiere ningún aparato especial (aparte del propio equipo de prueba). Asimismo, la sonda volante puede acceder a más puntos de la tarjeta porque se puede mover a cualquier posición y la sonda puede acceder a la parte superior de la tarjeta donde están montados los componentes. Una desventaja del método de la sonda volante es que resulta más lento que la cama de pinchos, por lo que generalmente está limitado a la prueba de prototipos y a las producciones pequeñas.



(a) Movimiento según 3 ejes



(b) Movimiento de punto a punto

FIGURA 11.74 Prueba de una tarjeta de circuito mediante sonda volante.

Pruebas de exploración de contorno

Las limitaciones en el acceso a los puntos de prueba condujeron al concepto de colocar dichos puntos de prueba dentro de los propios circuitos integrados. La mayoría de los dispositivos CPLD y FPGA incluyen lógica de exploración de contorno como parte de su estructura interna; esa lógica es independiente de la funcionalidad de la lógica programada en el dispositivo. Estos dispositivos son compatibles con el estándar JTAG.

Entre la lógica programable y cada uno de los pines de entrada y de salida del dispositivo se coloca un circuito, conocido con el nombre de celda de exploración de contorno, como se muestra en la Figura 11.75. Básicamente, estas celdas son celdas de memoria que almacenan un 1 o un 0. Las celdas conectadas a las entradas de la lógica programable se denominan celdas de entrada, mientras que aquellas conectadas a las salidas de la lógica programable se denominan celdas de salida. Las pruebas de ***exploración de contorno*** están basadas en el estándar JTAG (IEEE Std. 1149.1). Las cuatro entradas y salidas JTAG, denominadas TDI (*Test Data In*), TDO (*Test Data Out*), TCK (*Test Clock*)y TMS (*Test Mode Select*), se conocen con el nombre de puertos de acceso de prueba (TAP, *Test Access Port*).

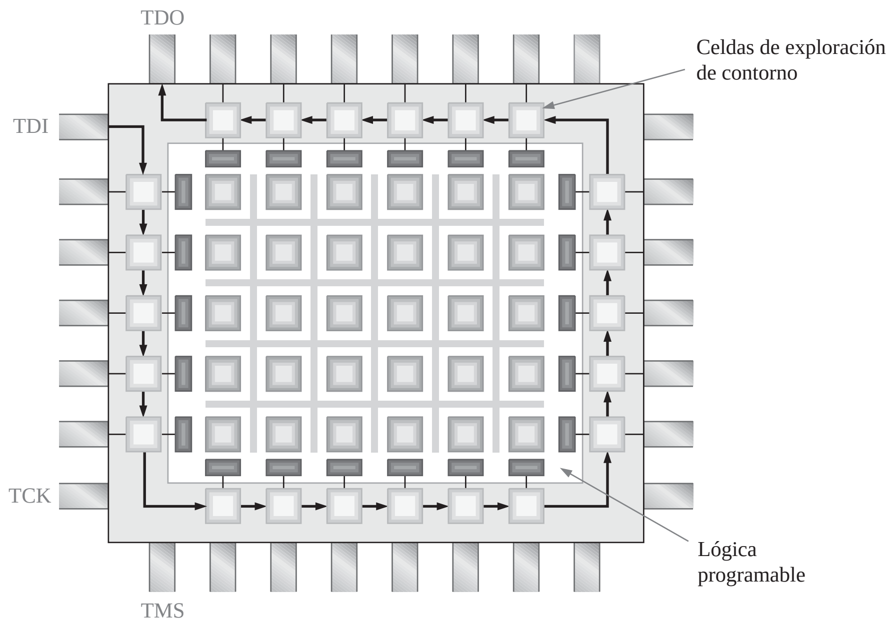


FIGURA 11.75 Concepto en que se basa la lógica de exploración de contorno en un dispositivo programable.

Modo Intest Cuando se utilizan las celdas de exploración de contorno para comprobar la funcionalidad interna del dispositivo, el modo de prueba se denomina *Intest.* El concepto básico de la exploración de contorno utilizando *Intest* es el siguiente: se introduce mediante desplazamiento a través del pin TDI un patrón de 1s y 0s controlado por software y ese patrón se coloca en las entradas de la lógica programable. Como resultado de la aplicación de estos bits de entrada, la lógica producirá en respuesta una serie de bits de salida. Los bits de salida resultante se extraen entonces por desplazamiento a través del pin TDO comprobándose si existe algún error. Una salida incorrecta indicará, por supuesto, un fallo en la lógica programada, en las celdas de E/S o en las celdas de exploración de contorno.

La Figura 11.76 muestra un patrón *Intest* de valor 1011 para la exploración de contorno de un circuito lógico AND-OR que ha sido programado en un dispositivo. Con dieciséis combinaciones de cuatro bits TDI se podría comprobar el circuito en todos los posibles estados de acuerdo con la lista de la Tabla 11.1. Las com-

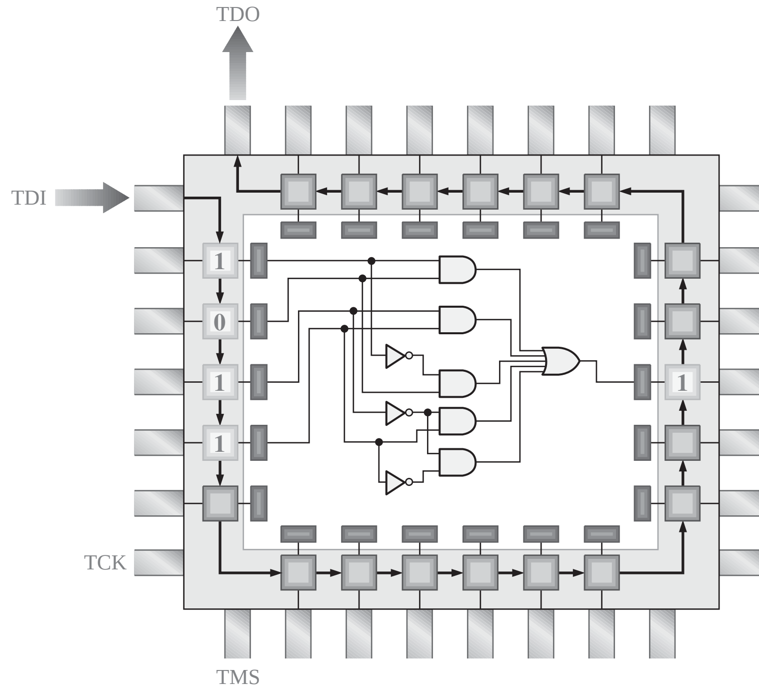


FIGURA 11.76 Ejemplo de patrón de bits en una prueba *Intest* mediante exploración de contorno para la comprobación de la lógica interna.

binaciones de 4 bits se desplazan en serie para introducirlas en las celdas de exploración de contorno, y la correspondiente salida se desplaza hacia fuera a través del pin TDO con el fin de comprobarla. Este proceso se controla mediante software de prueba de exploración de contorno.

Modo Extest Cuando se emplean celdas de exploración de contorno para probar las conexiones externas del dispositivo, además de la funcionalidad interna, el modo de prueba se denomina *Extest*. El concepto básico de exploración de contorno mediante *Extest* es el siguiente: se aplica a los pines de entrada del dispositivo un patrón de 1s y 0s definido por software y ese patrón se introduce en las celdas de entrada. Como resultado de la aplicación de estos bits de entrada, la lógica producirá como respuesta una serie de bits de salida. Entonces, los bits de salida resultantes se extraen a través de los pines de salida del dispositivo con el fin de comprobar la existencia de errores. Por supuesto, una salida incorrecta indicará que hay un fallo en las conexiones de los pines de entrada o de salida, que hay un fallo en las interconexiones de la tarjeta, que el dispositivo está fallando o que la instalación del dispositivo se ha realizado de manera inapropiada. Por supuesto, también pueden detectarse en el modo *Extest* algunos fallos internos. Por ejemplo, los fallos en la celdas de exploración de contorno, en las celdas de E/S o algunos fallos en la lógica programada en el dispositivo producirán una salida incorrecta. La Figura 11.77 muestra un ejemplo de prueba *Extest* de exploración de contorno que comprueba las cuatro entradas y la salida del circuito lógico.

Si se detecta un fallo en el modo *Extest*, puede tratarse de un fallo externo (una conexión de pin errónea) o interno (una conexión incorrecta, una celda de exploración de contorno erróneo o un elemento lógico incorrecto). Por tanto, para poder aislar un fallo detectado mediante una prueba *Extest* debe ejecutarse a continuación una prueba *Intest*. Si ambas pruebas muestran la existencia de fallo, entonces éste será interno al dispositivo.

En el modo *Extest*, es necesario comprobar los contactos existentes con los pines de entrada y de salida del dispositivo. Estos pines deben estar disponibles en algún conector que permita acceder a la tarjeta del

TDI	TDO
0000	1
0001	1
0010	0
0011	1
0100	1
0101	1
0110	1
0111	1
1000	1
1001	1
1010	0
1011	1
1100	1
1101	1
1110	1
1111	1

TABLA 11.1 Patrón de bits de prueba de exploración de contorno para el dispositivo programado en la Figura 11.76.

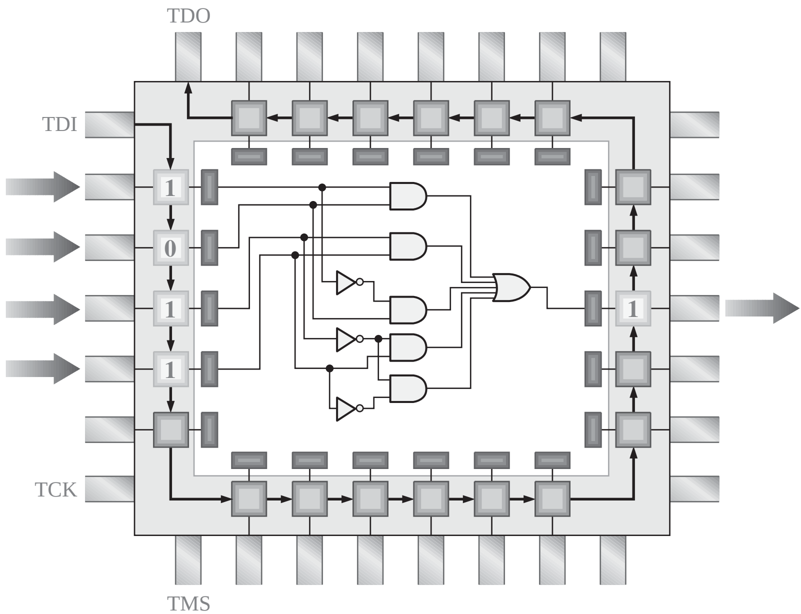


FIGURA 11.77 Ejemplo de patrón de bits en la prueba *Extest* de exploración de contorno para detectar fallos externos.

circuito o en una serie de conexiones de prueba, para poder realizar las comprobaciones mediante equipos de prueba automática. Los pines que no se conecten a través de un conector de tarjeta JTAG pueden probarse en

un punto de prueba utilizando el método de la cama de pinchos o de la sonda volante. Estas técnicas combinadas se ilustran en la Figura 11.78.

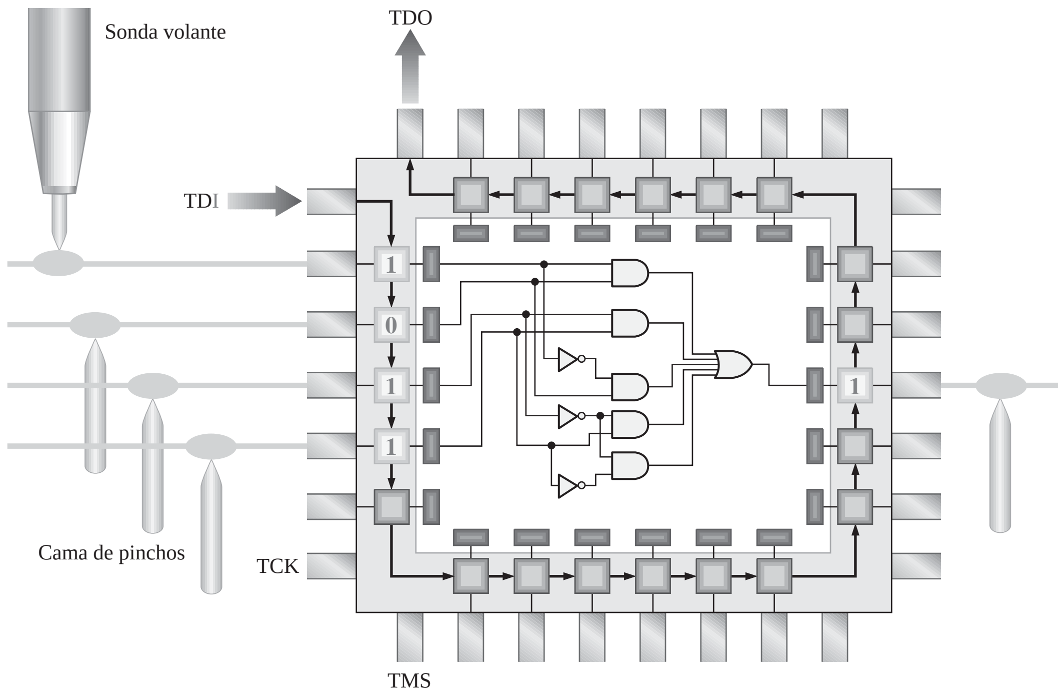


FIGURA 11.78 Combinación de pruebas de exploración de contorno, de cama de pinchos y de sonda volante.

Lenguaje BSDL (Boundary Scan Description Language) Este software de pruebas forma parte del estándar JTAG IEEE 1149.1 y utiliza VHDL para describir cómo implementar la lógica de exploración de contorno en un dispositivo específico y cómo opera esa lógica. BSDL proporciona un formato de datos estándar para describir el modo en que se implementa el estándar IEEE 1149.1 en un dispositivo compatible JTAG. Al utilizar herramientas software de pruebas de exploración de contorno que soporten BSDL, normalmente es el fabricante del dispositivo el que suministra el código BSDL.

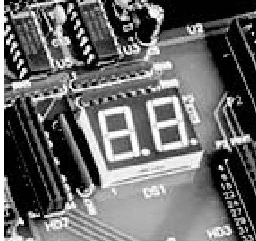
Cada dispositivo que contiene lógica de exploración de contorno dedicada está soportado por un archivo que describe dicho dispositivo concreto. Entre las cosas que se describen en el archivo BSDL están el tipo de dispositivo y una serie de descripciones de puertos que indican los pines de E/S y los pines del puerto de acceso de pruebas (TAP), indicando si se trata de entradas, salidas o pines bidireccionales. BSDL también establece la correspondencia entre las señales lógicas y los pines físicos y describe la arquitectura de la lógica de exploración de contorno contenida en el dispositivo. Utilizando BSDL, puede definirse un patrón de bits de prueba para comprobar el dispositivo.

REVISIÓN DE LA SECCIÓN 11.10

1. Describir el concepto básico de las pruebas de cama de pinchos para tarjetas electrónicas.
2. ¿Qué limitaciones tiene el método de la cama de pinchos?
3. ¿En qué se diferencia el método de prueba basado en sonda volante del método de la cama de pinchos?

REVISIÓN DE LA SECCIÓN 11.10

4. Explicar el concepto en que se basa la exploración de contorno.
5. ¿Cuáles son los dos modos de prueba de exploración de contorno?
6. Indicar cuatro señales JTAG que se utilicen en las pruebas de exploración de contorno.
7. ¿Qué es el lenguaje BSDL?



APLICACIÓN A LOS SISTEMAS DIGITALES

En esta aplicación de sistemas, aplicaremos las herramientas de desarrollo software genéricas, utilizando el procedimiento de introducción de esquemáticos para la lógica del decodificador BCD a 7-segmentos desarrollado en el Capítulo 4. Sólo vamos a mostrar aquí los pasos principales, adoptando un enfoque genérico para ilustrar los conceptos básicos.

La Figura 11.79 muestra los circuitos lógicos individuales para cada uno de los siete segmentos. El circuito lógico de cada segmento se introduce en un archivo independiente y luego se convierte en un símbolo de bloque. Una vez introducidos los archivos lógicos de los siete segmentos, se les combina en un único bloque, que formará el decodificador completo. También podríamos introducir de una sola vez los siete circuitos lógicos, para crear lo que se conoce como esquemático "plano". Sin embargo, utilizaremos el enfoque jerárquico para la introducción del diseño, con el fin de que la cantidad de lógica que aparezca en cada momento en la ventana del editor gráfico sea más manejable. Esta técnica resulta preferible cuando el esquemático es muy complejo y puede descomponerse en varias partes. Asimismo, en aquellas situaciones en las que haya varias personas trabajando en una serie de circuitos que posteriormente se combinarán para formar un circuito o sistema de mayor tamaño, el enfoque jerárquico resulta esencial.

Cada uno de los siete circuitos lógicos se introduce individualmente, se convierte a símbolo de bloque y se guarda. Una vez introducidos los siete circuitos, cada símbolo de bloque se coloca en la pantalla de introducción de esquemas. Todos los símbolos de bloque se conectan a continuación a las entradas y a las salidas. Recuerde que esto no es más que una descripción genérica, pero ilustra los conceptos básicos de algunas de las herramientas principales que podemos encontrar en casi todos los paquetes software, como el paquete Quartus II de Altera y el paquete ISE de Xilinx.

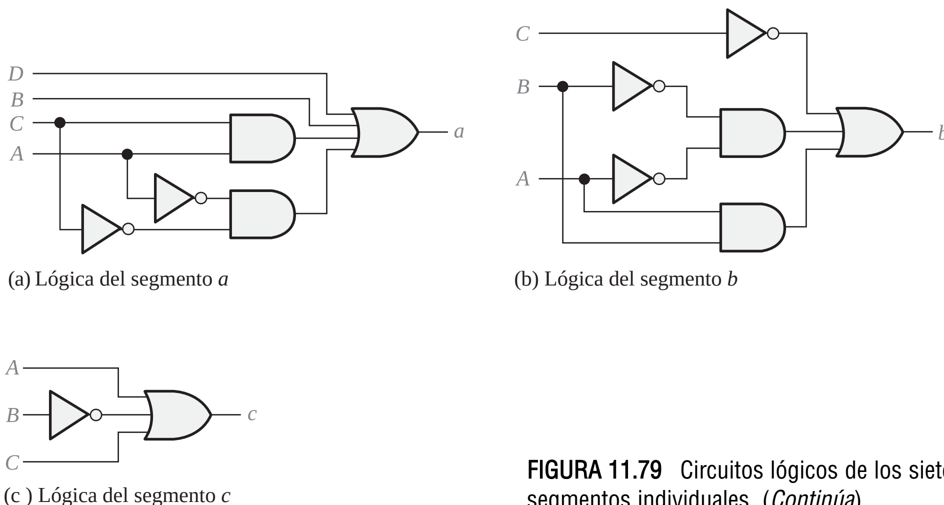


FIGURA 11.79 Circuitos lógicos de los siete segmentos individuales. (*Continúa*)

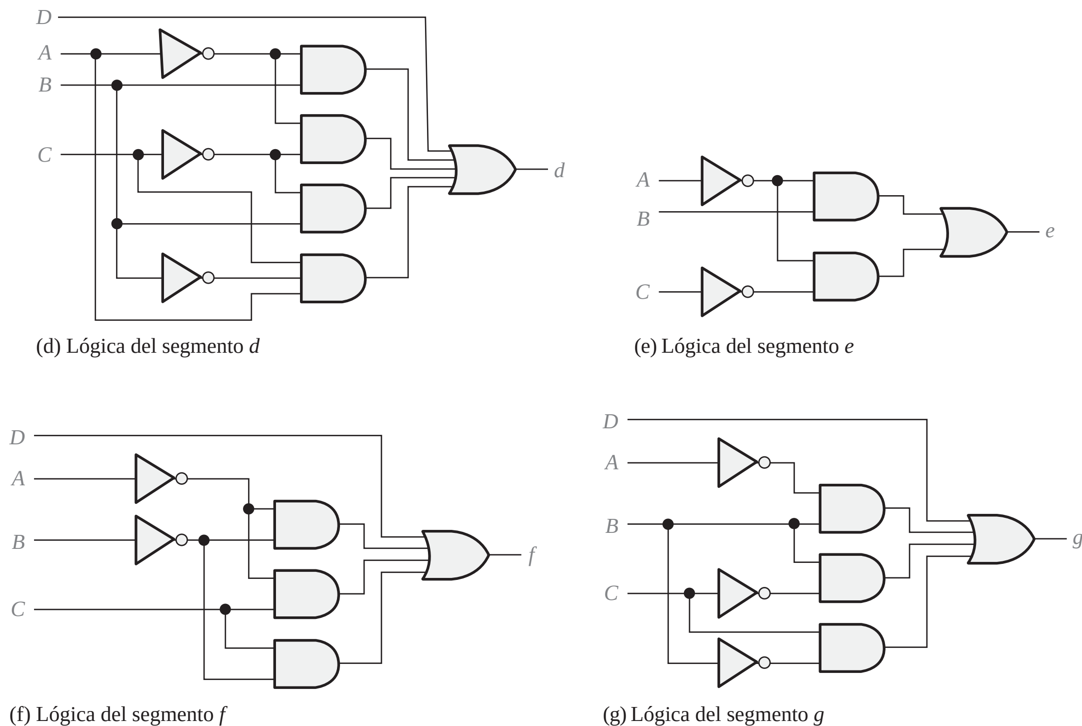


FIGURA 11.79 Circuitos lógicos de los siete segmentos individuales. (*Continuación*)

Introducción del diseño de la lógica del segmento *a*

Después de arrancar el software, iniciaremos un proyecto para la lógica de los 7-segmentos y abriremos la pantalla del editor gráfico. Para colocar en la pantalla los símbolos de las puertas lógicas, haga clic sobre el icono de la puerta como se indica en la Figura 11.80. Aparecerá una pantalla de símbolos en la que podremos seleccionar las puertas que deseemos entre todas las contenidas en la biblioteca software. Las puertas lógicas se denomina ***primitivas*** y pueden seleccionarse de una lista, bajo un encabezado genérico como el que se muestra en la Figura 11.80 y que agrupa a todas las primitivas.

Para el segmento *a*, seleccionamos dos instancias de la puerta AND de 2 entradas y las colocamos en la pantalla tal como se muestra. A continuación, seleccionamos una puerta OR de 4 entradas y colocamos una instancia en la pantalla. Finalmente, elegimos el inversor (NOT) y colocamos en la pantalla dos instancias del mismo.

A continuación, en la ventana de símbolos seleccionamos la sección de la biblioteca correspondiente a los pines. Podemos especificar un pin de entrada o un pin de salida, como se muestra en la Figura 11.81. Para este circuito concreto, colocamos cuatro pines de entrada y un pin de salida en la pantalla del editor gráfico, como se muestra en la Figura 11.82.

En la Figura 11.83 se muestra el esquemático completo correspondiente a la lógica del segmento *a*.

Compilación del diseño

Después de introducir el diseño, el siguiente paso consiste en compilarlo. El compilador es una herramienta software que se encarga de gestionar el proceso de flujo de diseño. La ***herramienta de encaje*** del compilador selecciona las interconexiones óptimas, las asignaciones de pines y las asignaciones óptimas de celdas lógicas para hacer que encaje un diseño en el dispositivo de destino seleccionado. Generalmente, se abrirá un cuadro de diálogo del compilador que, al arrancar, indicará el progreso realizado mediante un gráfico de barras y un porcentaje de terminación, como se muestra en la Figura 11.84. Aparecerá una indicación que nos informará sobre si el proceso de compilación ha tenido éxito o no.

Graphic Editor

File Edit View Project Assignments Processing Tools Window

Haga clic en el icono de puerta para abrir la ventana de selección de símbolos.

En el cuadro Libraries (bibliotecas), seleccione las primitivas y resalte la puerta que desee colocar en el editor gráfico. Haga clic en OK. A continuación, coloque tantas instancias como sean necesarias.

Continúe seleccionando las puertas necesarias y colóquelas en el editor gráfico.

Symbol

Libraries

Primitives
and2
and3
and4
or2
or3
or4
not

OK Cancel

FIGURA 11.80 Ilustración de la selección y colocación de los símbolos lógicos en el editor gráfico.

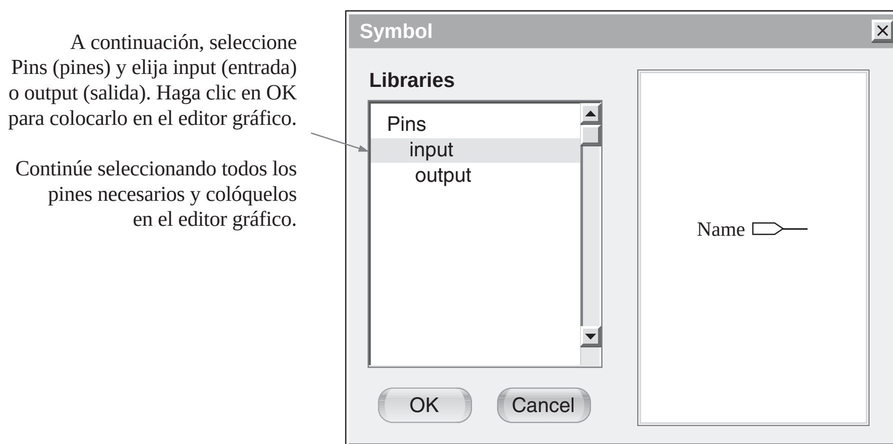


FIGURA 11.81 Selección de los pines de entrada y de salida en la ventana de símbolos (Symbol).

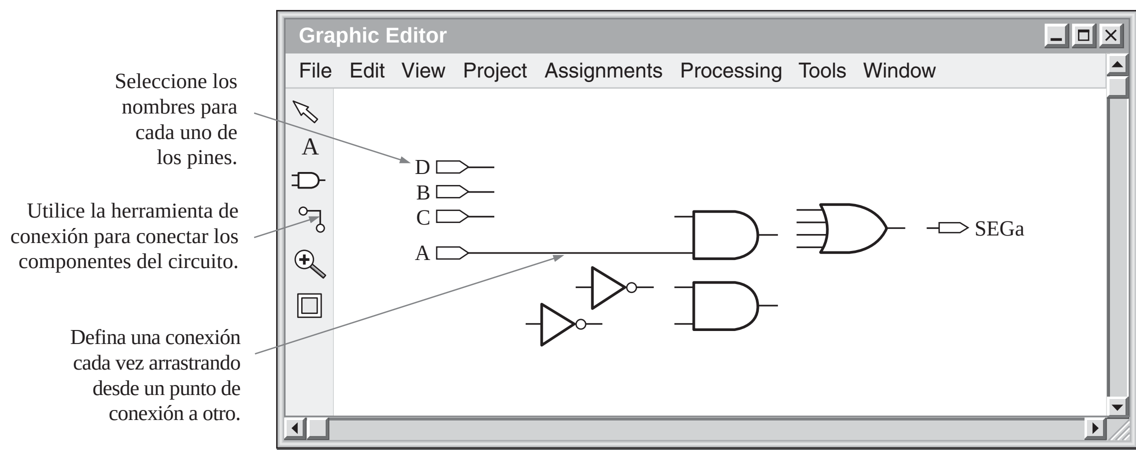


FIGURA 11.82 Colocación de los pines y realización de las conexiones del circuito en el editor gráfico.

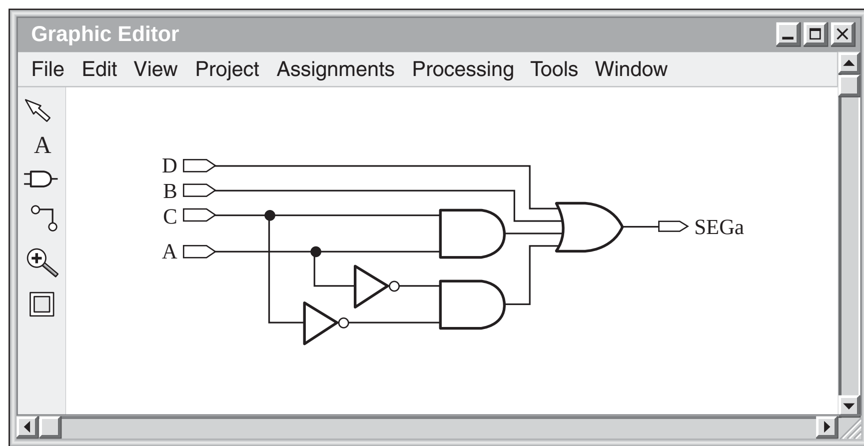


FIGURA 11.83 Esquemático completo para la lógica del segmento *a*.

Simulación funcional

La mayor parte de los paquetes de software disponen, al menos, de dos tipos de herramienta de simulación: simulación funcional y simulación de temporización. La ***simulación funcional*** verifica la funcionalidad del circuito lógico y debe llevarse a cabo en cuanto se haya compilado con éxito el circuito.

El primer paso para preparar la simulación consiste en especificar una señal como entrada o como salida y asignar un nombre. A continuación, se van creando formas de onda, seleccionando el intervalo de tiempo deseado y especificando el nivel ALTO (1) o BAJO (0). Podemos definir de esta forma los sucesivos niveles hasta completar cada forma de onda. Este proceso se repite para todas las demás

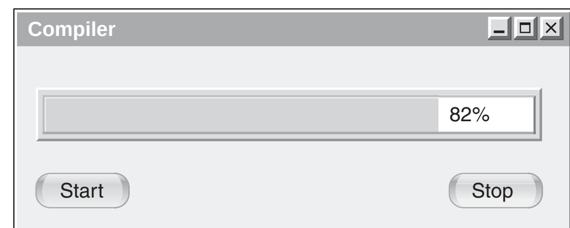


FIGURA 11.84 Cuadro de diálogo del compilador.

formas de onda de entrada, como se indica en la Figura 11.85.

Una vez especificadas todas las formas de onda de entrada, se inicia la simulación y se genera una forma de onda de salida, como se muestra en la Figura 11.85 para la lógica del segmento *a*.

Creación del símbolo del bloque

Si la simulación tiene éxito, quiere decir que el circuito lógico funciona de la forma esperada desde el punto de vista funcional, es decir, que la lógica es correcta. El paso siguiente consiste en convertir el esquemático lógico en un símbolo lógico, como se ilustra en la Figura 11.86, y guardarlo para un uso posterior. Una vez que se ha guardado el símbolo de bloque, podemos acceder a él para utilizarlo en el diseño final.

Introducción del diseño para los segmentos *b* hasta *g*

Repetimos para cada uno de los circuitos lógicos de los otros seis segmentos el mismo procedimiento general que

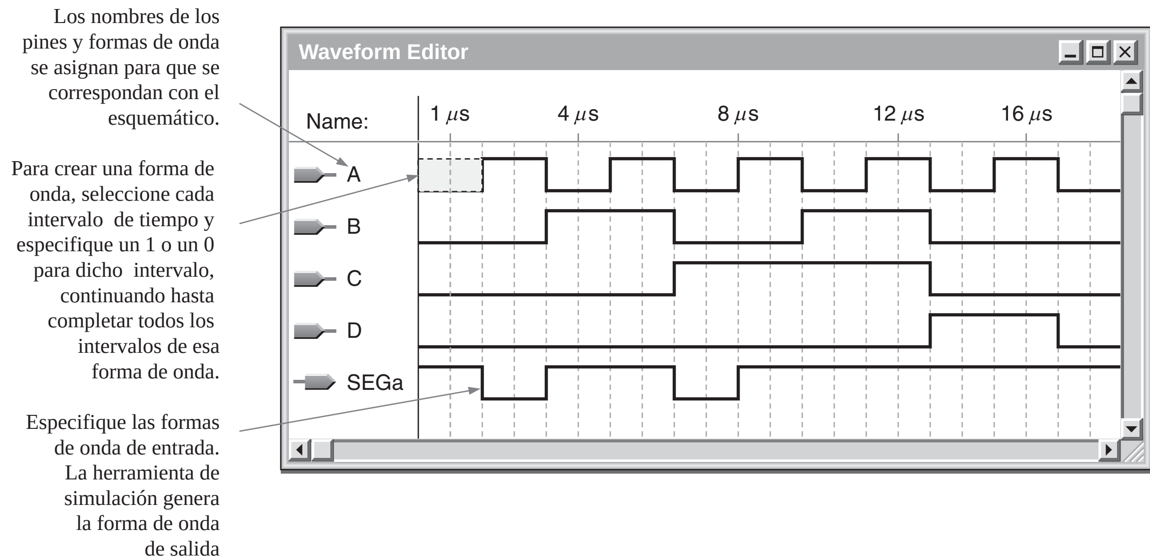


FIGURA 11.85 Formas de onda de entrada y forma de onda de salida resultante para la lógica correspondiente al segmento *a*.

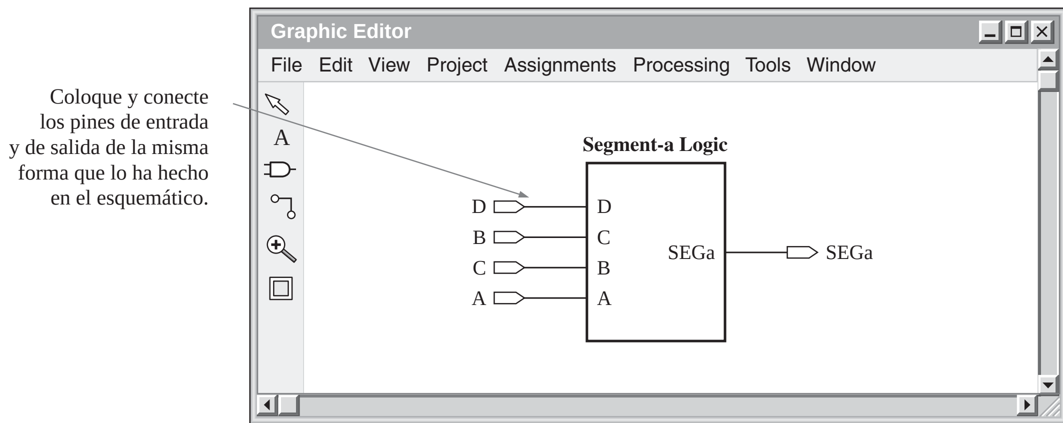


FIGURA 11.86 Esquemático de la lógica del segmento *a* convertido en un símbolo de bloque.

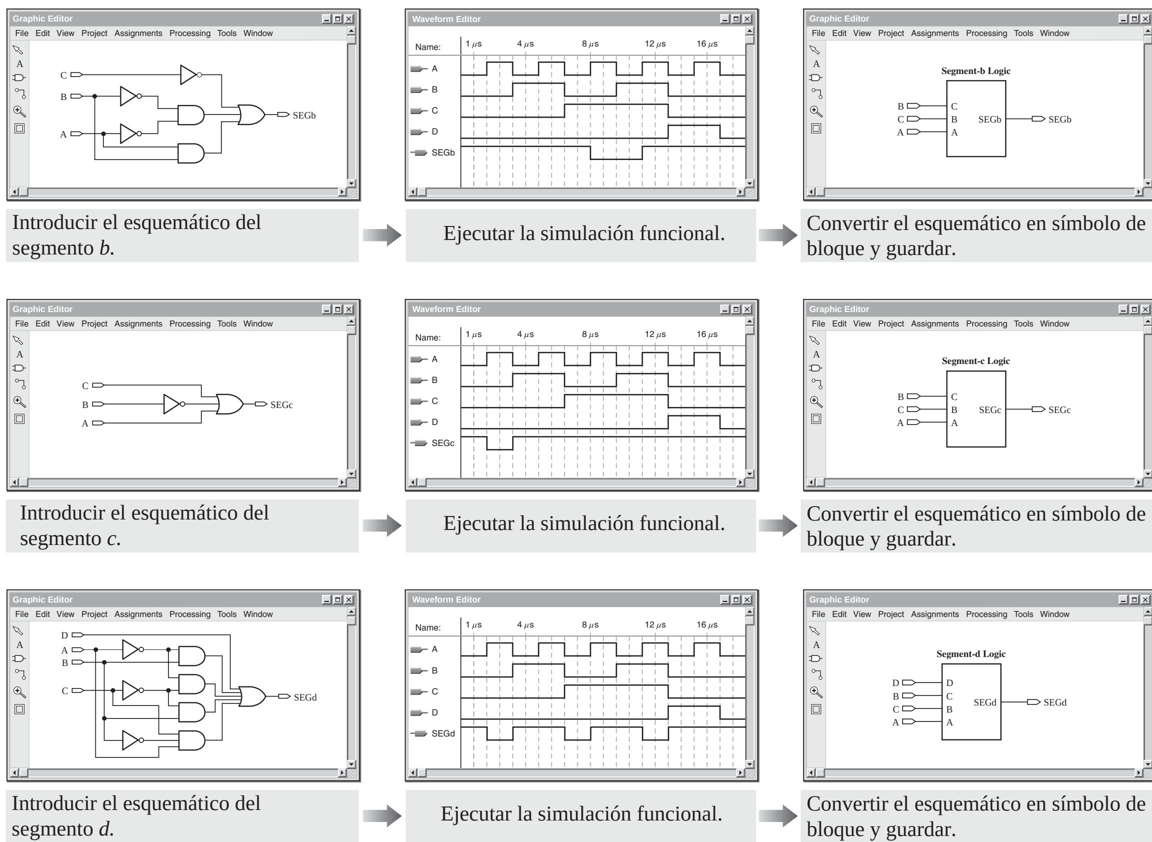


FIGURA 11.87 Pantallas correspondientes a la lógica de los segmentos *b*, *c* y *d*.

hemos utilizado para introducir, simular y guardar en forma de símbolo de bloque la lógica del segmento *a*. La Figura 11.87 muestra las pantallas para la lógica correspondientes al segmento *b*, el segmento *c* y el segmento *d*.

La Figura 11.88 muestra las pantallas para la lógica del segmento *e*, la lógica del segmento *f* y la lógica del segmento *g*. Dejamos como ejercicio para el lector las simulaciones correspondientes a los segmentos *e, f* y *g*.

Diagrama de bloques final

Hemos guardado todos los símbolos de bloque correspondientes a la lógica de los segmentos y ahora podemos utilizarlos para construir la lógica completa de 7 segmentos. Accedemos a los símbolos a través de la ventana de símbolos, igual que hicimos a la hora de seleccionar puertas lógicas. Para ello, abrimos el archivo en el que están almacenados, con lo que nos aparecerá la lista de símbolos, como se muestra en la Figura 11.89. Añadimos los pines de entrada y de salida y conectamos todos los símbolos de bloque, para formar la lógica completa de excitación de los segmentos.

Simulación temporal

Después de haber introducido el circuito completo en el editor gráfico, debemos compilarlo de la misma manera en que compilamos el circuito lógico correspondiente a cada uno de los segmentos. Después de completada la compilación, es necesario llevar a cabo una simulación temporal o simulación de temporización. La ***simulación de temporización*** tiene en cuenta los retardos de propagación de cada puerta del diseño, además de la funcionalidad del circuito. Las formas de onda de entrada se especifican de la misma forma que para la simulación funcional. En la Figura 11.90 se muestra el editor de formas de onda con una simulación temporal en la que no hay ningún problema aparente, como por ejemplo, impulsos de ruido (*glitches*). Si existiera algún impulso de ruido, sería necesario volver al diseño y tratar de corregir las condiciones

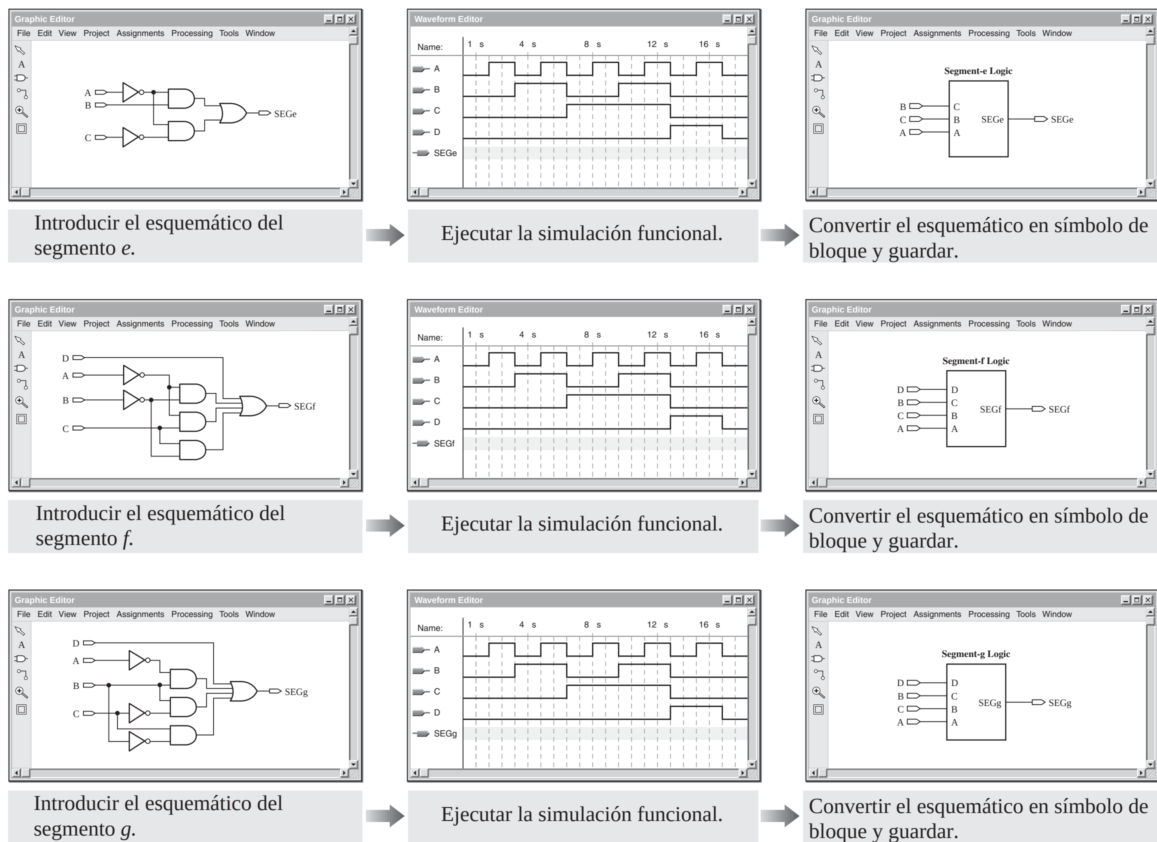


FIGURA 11.88 Pantallas correspondientes a la lógica de los segmentos *e*, *f* y *g*. Las formas de onda de salida para cada caso se completan como actividad.

que lo hacen posible, si es que ese *glitch* puede constituir un potencial problema.

Una simulación temporal adecuada significa que el circuito lógico funciona de la forma esperada tanto en términos de funcionalidad como de temporización. El siguiente paso consiste en convertir el diagrama con múltiples bloques en un único símbolo de bloque, como se ilustra en la Figura 11.91, para un posible uso futuro.

Programación del dispositivo de destino

Suponiendo que la simulación de temporización indique que no hay ningún problema, el siguiente paso consiste en programar el dispositivo de destino con la lógica de 7-segmentos. Inicie la secuencia de programación y seleccione la interfaz correspondiente, como por ejemplo JTAG. A continuación, seleccione el dispositivo de destino en la ventana de selección de dispositivo (Select Device), como se muestra en la Figura 11.92. Recuerde que, para un circuito con el tamaño de nuestro circuito de ejemplo, sólo se va a utilizar un porcentaje muy pequeño de un dispositivo programable tipo. En un único dispositivo PLD pueden programarse miles de circuitos con un número similar de puertas. A menudo, la limitación será el número de entradas y de salidas que hay disponibles en el encapsulado del dispositivo de destino. Los paquetes software permiten, generalmente, asignar a las entradas y salidas los correspondientes números de pin del dispositivo. Sin embargo, si decidimos no realizar ninguna asignación, la mayor parte de los paquetes software realizarán automáticamente esa asignación por nosotros y nos proporcionarán una lista de las asignaciones realizadas.

Una vez completa la selección de dispositivo, se descarga el diseño, como se indica en el mensaje de terminación de descarga de la Figura 11.93.

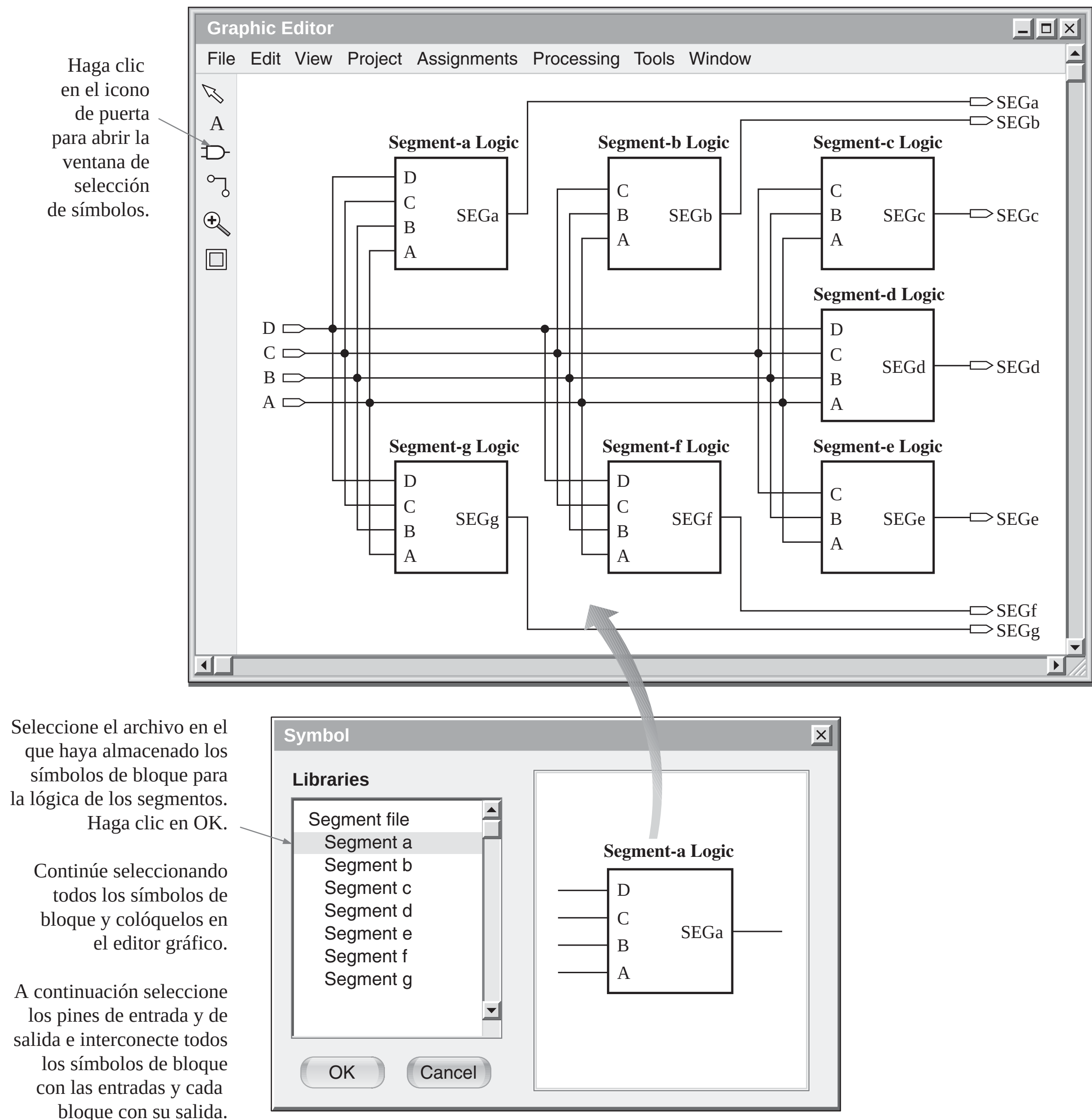


FIGURA 11.89 Selección, colocación e interconexión de la lógica de segmentos en el editor gráfico.

Pruebas dentro del circuito

Después de descargado el diseño en el dispositivo de destino, el siguiente paso suele ser la realización de pruebas hardware. Anteriormente, sólo hemos efectuado pruebas software para verificar la operación del circuito. Ahora, lo que haremos será conectar instrumentos reales al circuito montado en la tarjeta de desarrollo, aplicar señales de entrada y observar las señales de salida. El método que se utilice dependerá del tipo y de la complejidad de la lógica que haya sido descargada en el dispositivo, pero generalmente se empleará una fuente de señal (como por ejemplo un generador de funciones o un generador de patrones) para suministrar las entradas y un osciloscopio o analizador lógico para observar las señales de salida.

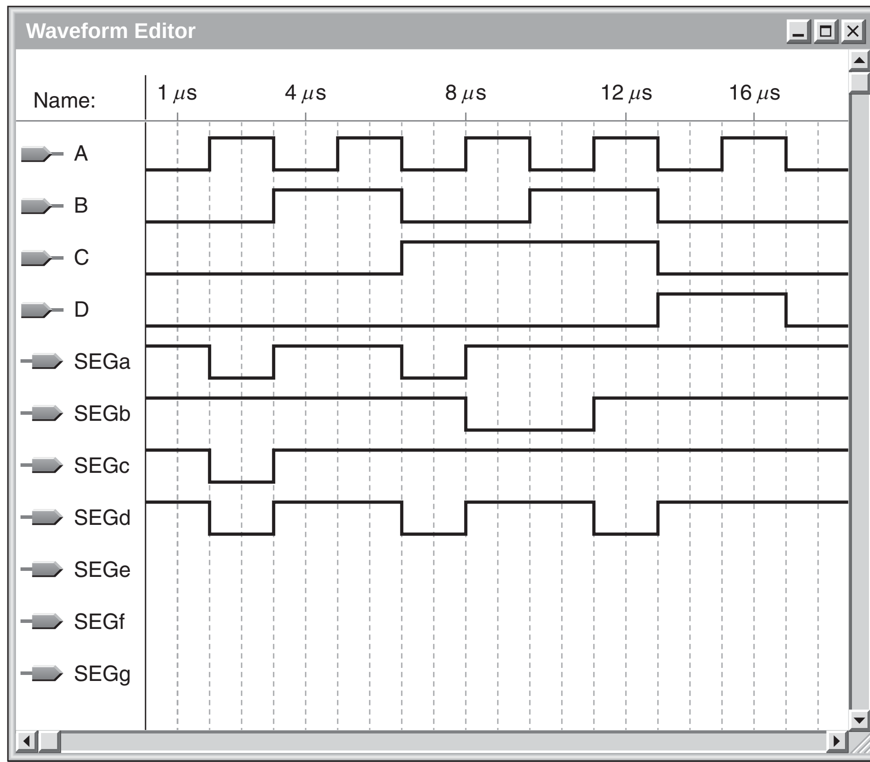


FIGURA 11.90 Simulación temporal ideal para la lógica completa de 7-segmentos de la Figura 11.89. Se deja como ejercicio para el lector el completar las formas de onda correspondientes a los segmentos *e*, *f* y *g*.

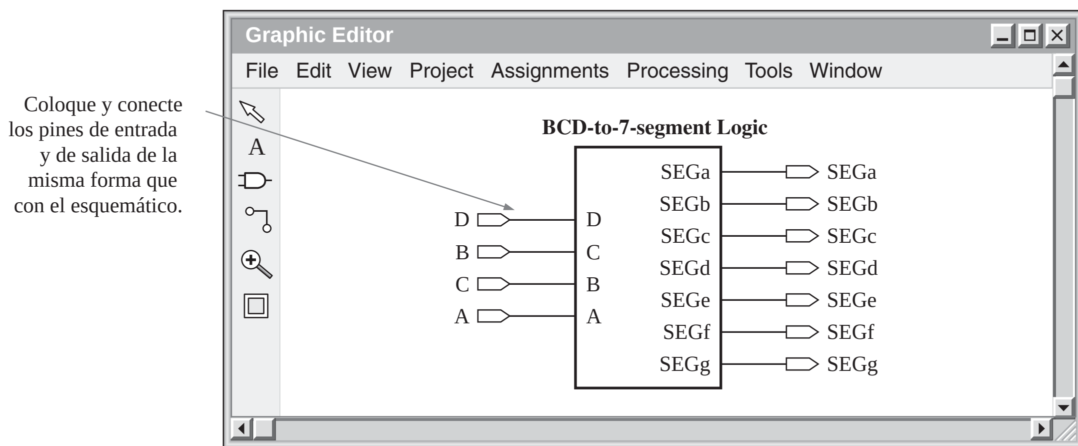


FIGURA 11.91 El decodificador completo BCD a 7-segmentos como un único símbolo de bloque.

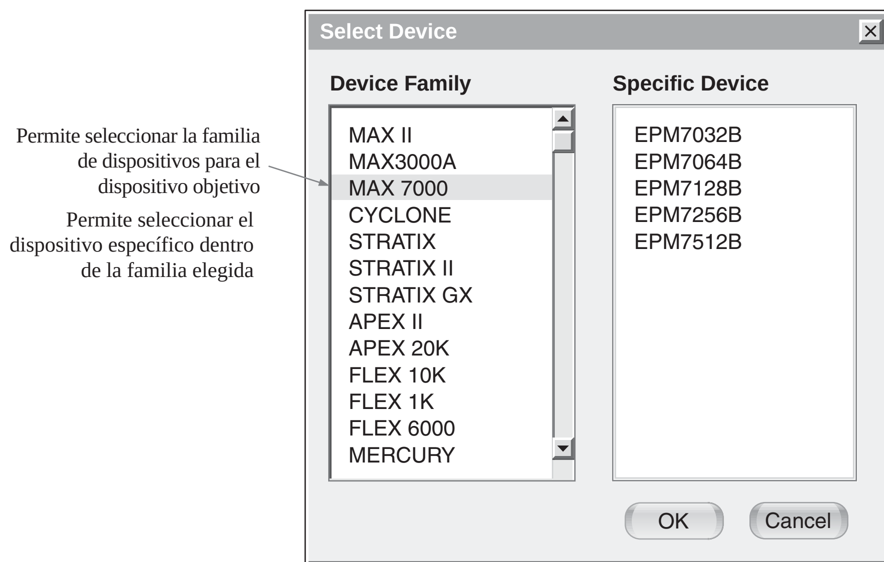


FIGURA 11.92 Selección del dispositivo de destino.

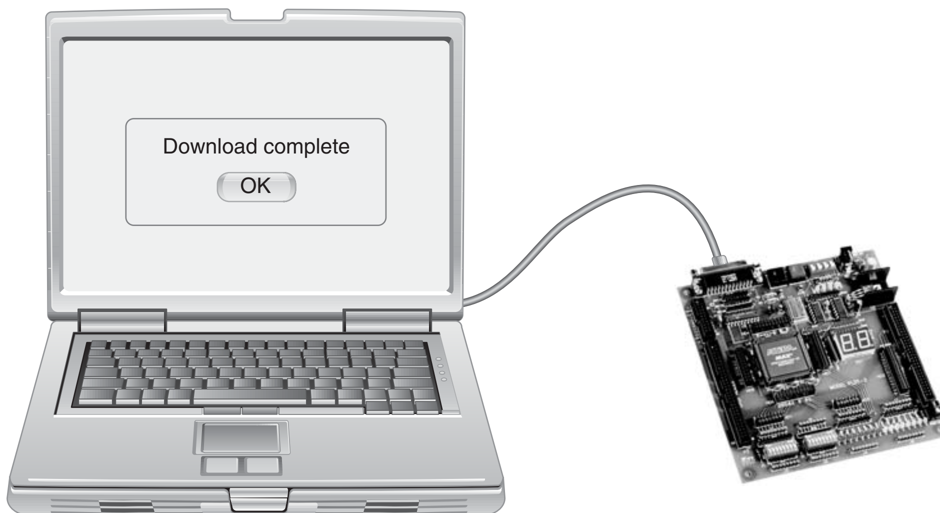


FIGURA 11.93 El diseño se ha descargado en el dispositivo de destino que se encuentra en una tarjeta de desarrollo.

En el caso de la lógica del decodificador de 7-segmentos que hemos descargado en el dispositivo de destino, podemos llevar a cabo una prueba simple utilizando los métodos disponibles en la propia tarjeta de desarrollo. Las entradas pueden conectarse a los conmutadores existentes en la tarjeta y las salidas pueden conectarse al display de 7-segmentos que la tarjeta incorpora. Después de conectar una fuente de alimentación, podemos ir introduciendo los distintos códigos BCD mediante los conmutadores y observar la salida en el display. La mayoría de las tarjetas de desarrollo disponen de este tipo de elementos, ilustrados en la Figura 11.94.

Prácticas de sistemas

- ***Actividad 1*** Verificar la corrección de la forma de onda de salida SEG*a* de la Figura 11.85.
- ***Actividad 2*** Verificar la corrección de las formas de onda de salida de los segmentos *b*, *c* y *d* de la Figura 11.87.
- ***Actividad 3*** Determinar las formas de onda de salida correctas para los segmentos *e*, *f* y *g* en las Figuras 11.88 y 11.90.

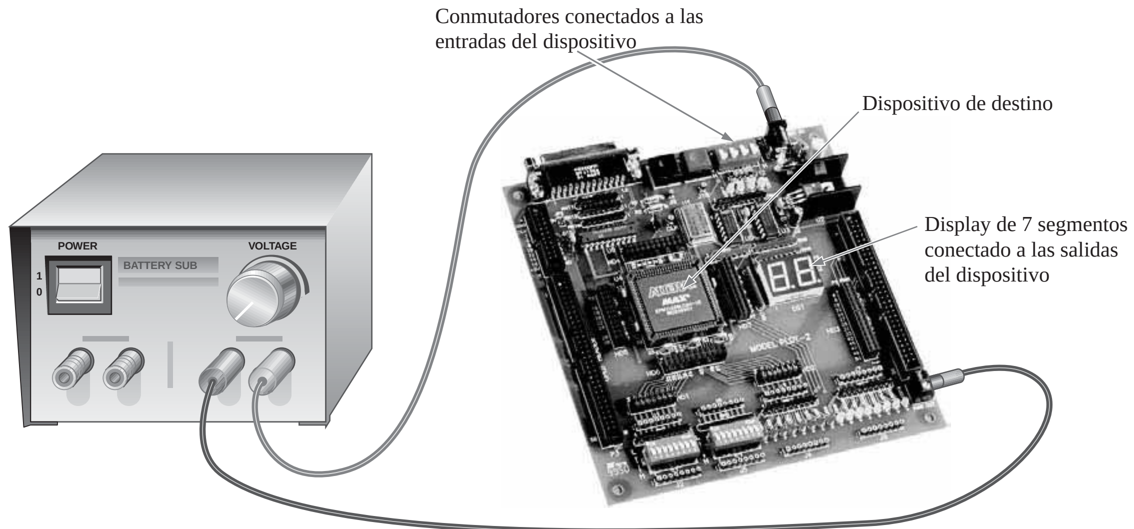


FIGURA 11.94 Ejemplo de dispositivo de destino montado en una tarjeta de desarrollo y que se prueba mediante los conmutadores y el display de 7-segmentos incluidos en la propia tarjeta.

RESUMEN

- Una PAL es un SPLD programable una sola vez y que está compuesto por una matriz programable de puertas AND y por una matriz fija de puertas OR.
- La estructura de una PAL permite implementar cualquier expresión lógica suma de productos con un número definido de variables.
- Una GAL es, esencialmente, una PAL que puede reprogramarse.
- En una PAL o GAL, cada macrocelda está compuesta generalmente de una puerta OR y cierta lógica de salida asociada.
- La PAL16V8 es un tipo común de dispositivo PAL.
- La GAL22V10 es un tipo común de dispositivo GAL.
- Un CPLD es un dispositivo lógico programable complejo que está compuesto, básicamente, de múltiples matrices SPLD con interconexiones programables.
- Cada matriz SPLD en un CPLD se denomina bloque de matriz lógica (LAB, *Logic Array Block*).
- MAX 7000 es una familia de dispositivos CPLD de Altera.

- En la familia de dispositivos CPLD MAX 7000, la densidad de los dispositivos va de 2 a 16 bloques LAB, dependiendo del dispositivo concreto de la serie. Cada bloque LAB tiene 16 macroceldas.
- Los dispositivos CPLD MAX II de Altera difieren enormemente de los dispositivos de la familia MAX 7000, por lo que se dice que estos dispositivos son dispositivos CPLD "post-macrocelda".
- Un CPLD MAX II utiliza tablas de consulta (LUT) en lugar de matrices AND/OR.
- La arquitectura de la familia de dispositivos CPLD CoolRunner II de Xilinx está basada en una estructura PLA en lugar de en una estructura PAL.
- La familia CoolRunner II contiene una serie de dispositivos CPLD que van desde 32 a 512 macroceldas.
- Una macrocelda se puede configurar en uno de dos modos: modo combinacional o modo registrado.
- Una FPGA (*Field-Programmable Gate Array*, matriz de puertas programable sobre el terreno) difiere en cuanto a arquitectura de un CPLD típico, y utiliza matrices de tipo PAL/PLA y tiene densidades mucho mayores que un CPLD.
- La mayoría de las FPGA utilizan tecnología de proceso basada en antifusible o en memoria SRAM.
- Cada bloque lógico configurable (CLB) en una FPGA está compuesto de múltiples módulos lógicos más pequeños y de una interconexión local programable que se utiliza para conectar los módulos lógicos dentro del bloque CLB.
- Los dispositivos FPGA están basados en una arquitectura de tipo LUT.
- Una LUT (*Look-Up Table*, tabla de consulta) es un tipo de memoria programable que se utiliza para generar funciones lógicas combinacionales de tipo suma de productos.
- Un módulo hardware es una parte de lógica integrada en una FPGA y proporcionada por el fabricante con el fin de implementar una función específica; el módulo hardware no puede reprogramarse.
- Un módulo software es una parte de lógica integrada en una FPGA que tiene ciertas características programables.
- Los diseños que son propiedad del fabricante se denominan propiedad intelectual (IP, *Intellectual Property*).
- Altera fabrica varias familias de dispositivos FPGA, incluyendo las familias Stratix II, Stratix, Cyclone y ACEX.
- Xilinx tiene dos líneas principales de dispositivos FPGA: Spartan y Virtex, existiendo diferentes familias dentro de cada una de las líneas.
- El proceso de programación se denomina, generalmente, flujo de diseño.
- El dispositivo que se está programando se suele denominar dispositivo de destino.
- En los paquetes software para lógica programable, las operaciones se controlan mediante un programa de aplicación denominado compilador.
- Durante la descarga, se genera un flujo de bits que representa el diseño final, y ese flujo de bits se envía al dispositivo de destino para configurarlo automáticamente.
- El método de cama de pinchos fue una de las primeras técnicas para la realización de pruebas automatizadas en las tarjetas de circuito.
- Otro método de prueba de tarjetas de circuito es el que se denomina método de la sonda volante.
- Un método para comprobar internamente un dispositivo programable es la denomina exploración de contorno, que se basa en el estándar JTAG (IEEE Std. 1149.1).
- La lógica de exploración de contorno en un dispositivo CPLD está compuesta por un registro de exploración de contorno, un registro de puenteo, un registro de instrucción y un puerto de acceso de pruebas (TAP).

PALABRAS CLAVE

Las palabras clave y otros términos que se han resaltado en negrita se encuentran en el glosario final del libro.

Cama de pinchos Un método para la prueba automatizada de tarjetas de circuito, según el cual la tarjeta se monta sobre un utillaje que recuerda una cama de pinchos y que hace contacto con los puntos de prueba.
CLB *Configurable Logic Block*, bloque lógico configurable. Una unidad de lógica en una FPGA que está compuesta de múltiples bloques lógicos más pequeños y de una interconexión programable local que se emplea para conectar entre sí los módulos lógicos que componen el CLB.
Compilador Un programa de aplicación que forma parte de los paquetes software de desarrollo y que controla la operación del software.
CPLD Un dispositivo programable lógico complejo que está compuesto de múltiples matrices SPLD con interconexiones programables.
Descarga El paso final en un flujo de diseño mediante el que el diseño final se implementa en el dispositivo de destino.
Dispositivo de destino El dispositivo lógico programable que se quiere programar.
Exploración de contorno Un método para probar internamente un dispositivo PLD y que está basado en el estándar JTAG (IEEE Std. 1149.1).
Flujo de diseño El proceso de secuencia de operaciones llevado a cabo para programar un dispositivo de destino.
FPGA *Field Programmable Gate Array*, matriz de puertas programable sobre el terreno: un dispositivo lógico programable que utiliza una carga LUT como elemento lógico básico y que emplea, generalmente, tecnología de proceso basada en antifusible o basada en SRAM.
GAL Un tipo reprogramable de dispositivo SPLD, que es similar a una PAL salvo porque utiliza una tecnología de proceso reprogramable, como por ejemplo celdas EEPROM (E^2CMOS) en lugar de fusibles.
Herramienta de encaje Una herramienta del software de compilación que selecciona las interconexiones óptimas, las asignaciones óptimas y las asignaciones óptimas de celdas lógicas con el fin de encajar un diseño dentro del dispositivo de destino seleccionado.
Introducción de texto Un método de introducir un diseño lógico dentro del software utilizando un lenguaje de descripción hardware (HDL, *Hardware Description Language*).
Introducción de esquemáticos Un método de introducir un diseño lógico dentro del software utilizando símbolos esquemáticos.
LAB *Logic Array Block*, bloque de matriz lógica: una matriz SPLD dentro de un dispositivo CPLD.
LUT *Look-Up Table*, tabla de consulta: un tipo de memoria que puede programarse para generar funciones suma de productos.
Macrocelda Parte de una PAL, GAL o CPLD, que está compuesta, generalmente, por una puerta OR y cierta lógica de salida asociada.
PAL Un tipo de dispositivo CPLD programable una única vez que está compuesta por una matriz programable de puertas AND que se conecta a una matriz fija de puertas OR.
Primitiva Un elemento lógico básico tal como una puerta, un flip-flop, un pin de E/S, una conexión de masa o una conexión V_{CC}.
Propiedad intelectual (IP) Diseños que son propiedad de un fabricante de dispositivos lógicos programables.
Registrada, lógica Un modo de operación de una macrocelda en el que se emplea un flip-flop.
Simulación de temporización Un proceso software que utiliza la información sobre los retardos de propagación y los datos de la lista de interconexiones para comprobar tanto la operación lógica como la temporización de caso peor de un diseño.
Simulación funcional Un proceso software que comprueba la operación lógica y funcional de un diseño.

Sonda volante Un método para la prueba automatizada de tarjetas de circuito mediante el cual se mueven una o más sondas de un lugar a otro con el fin de hacer contacto con los puntos de prueba.

AUTOTEST

Las respuestas se encuentran al final del capítulo.

1. Dos tipos de SPLD son:
(a) CPLD y PAL **(b)** PAL y FPGA
(c) PAL y GAL **(d)** GAL y SRAM

2. Una PAL está compuesta por:
(a) una matriz AND programable y una matriz OR programable
(b) una matriz AND programable y una matriz OR fija
(c) una matriz AND fija y una matriz OR programable
(d) una matriz AND/OR fija

3. Una macrocelda está compuesta por:
(a) una puerta OR fija y otra lógica asociada
(b) una matriz OR programable y otra lógica asociada
(c) una matriz AND fija y otra lógica asociada
(d) una matriz AND/OR fija con un flip-flop

4. El 16V8 es un tipo de:
(a) CPLD **(b)** GAL **(c)** PAL **(d)** FPGA

5. La estructura AND/OR básica de los dispositivos SPLD y CPLD permite implementar expresiones booleanas de tipo:
(a) Producto de sumas **(b)** Suma de productos
(c) Producto de complementos **(d)** Suma de complementos

6. El término *LAB* quiere decir:
(a) *Logic AND Block* (Bloque AND lógico)
(b) *Logic Array Block* (Bloque de matriz lógica)
(c) *Last Asserted Bit* (Último bit activado)
(d) *Logic Assembly Block* (Bloque de conjunto lógico)

7. MAX 7000 es:
(a) Una familia de dispositivos CPLD **(b)** Una familia de dispositivos SPLD
(c) Una familia de dispositivos FPGA **(d)** Un tipo de software

8. CoolRunner es:
(a) Una familia de dispositivos CPLD **(b)** Una familia de dispositivos SPLD
(c) Una familia de dispositivos FPGA **(d)** Un tipo de software

9. Dos modos de operación de una macrocelda son:
(a) entrada y salida **(b)** registrada y secuencial
(c) combinacional y registrado **(d)** paralelo y compartido

10. Cuando una macrocelda se configura para generar una función suma de productos, decimos que está en modo:
(a) combinacional **(b)** paralelo
(c) registrado **(d)** compartido

11. Una macrocelda típica consta de:

(a) puertas, multiplexores y un flip-flop

(b) puertas y un registro de desplazamiento

(c) un contador de código Gray

(d) una matriz lógica fija

12. Dependiendo de la complejidad de sus bloques lógicos configurables (bloques CLB), una FPGA puede clasificarse como:

(a) volátil o no volátil

(b) programable o reprogramable

(c) de granularidad fina o de garnularidad gruesa

(d) de plataforma o integrada.

13. Las FPGA no volátiles generalmente están basadas en:

(a) tecnología de fusibles

(b) tecnología de antifusibles

(c) tecnología EEPROM

(d) tecnología SRAM

14. Una FPGA con una función lógica integrada que no puede programarse se dice que es:

(a) no volátil **(b)** de plataforma

(c) un módulo hardware **(d)** un módulo software

15. Los diseños de módulos hardware suelen ser desarrollados por el fabricante de la FPGA y son propiedad del mismo. Estos diseños de denominan:

(a) propiedad intelectual **(b)** lógica propietaria

(c) diseños personalizados **(d)** estándares IEEE

16. Para la introducción de texto de un diseño lógico,

(a) hay que emplear símbolos lógicos

(b) hay que emplear un lenguaje HDL

(c) sólo se utiliza álgebra booleana

(d) hay que emplear un código especial

17. En una simulación funcional, el usuario tiene que especificar:

(a) El dispositivo de destino específico **(b)** La forma de onda de salida

(c) Las formas de onda de entrada **(d)** El código HDL

18. La salida final de la fase de síntesis de un flujo de diseño es:

(a) La lista de interconexiones **(b)** El flujo de bits

(c) La simulación de temporización **(d)** Los números de pines del dispositivo

19. EDIF significa:

(a) *Electronic Device Interchange Format* (Formato de intercambio de dispositivos electrónicos)

(b) *Electrical Design Integrated Fixture* (Aparejo integrado para diseño eléctrico)

(c) *Electrically Destructive Input Function* (Función de entrada eléctricamente destructiva)

(d) *Electronic Design Interchange Format* (Formato de intercambio de diseños electrónicos)

20. El puerto TAP de la técnica de exploración de contorno significa:

(a) *Test Access Point* (Punto de acceso de prueba)

(b) *Test Array Port* (Puerto de matriz de prueba)

(c) *Test Access Port* (Puerto de acceso de prueba)
(d) *Terminal Access Path* (Ruta de acceso a terminal)

21. Una celda típica de exploración de contorno contiene:
(a) sólo flip-flops
(b) flip-flips y lógica de multiplexación
(c) latches y flip-flops
(d) latches y un codificador

22. Un método automatizado para probar tarjetas de circuito que usa un aparejo con múltiples contactos fijos con puntos de prueba de la tarjeta se denomina:
(a) tradicional (b) de sonda volante
(c) de cama de pinchos (d) de exploración de contorno

23. Un método automatizado para probar tarjetas de circuito donde se utiliza un contacto móvil con los puntos de prueba se denomina
(a) tradicional (b) de sonda volante
(c) de cama de pinchos (d) de exploración de contorno

24. El estándar JTAG tiene las siguientes entradas y salidas:
(a) Intest, Extest, TDI, TDO (b) TDI, TDO, TCK, TMS
(c) ENT, CLK, SHF, CLR (d) TCK, TMS, TMO, TLF

25. El acrónimo BSL significa:
(a) *Board Standar Digital Logic* (Lógica digital estándar de tarjetas)
(b) *Boundary Scan Down Load* (Descarga de exploración de contorno)
(c) *Bistable Logic Latch* (Latch digital biestable)
(d) *Boundary Scan Description Language* (Lenguaje de descripción para exploración de contorno)

PROBLEMAS

Las respuestas a los problemas impares se encuentran al final del libro.

SECCIÓN 11.1 Lógica programable: SPLD y CPLD

1. Determinar la expresión de salida booleana para la matriz PAL simple mostrada en la Figura 11.95. Las X representan enlaces conectados.

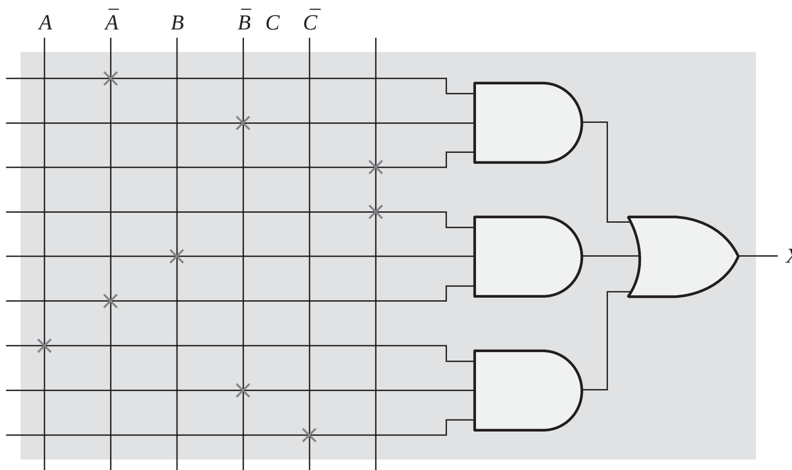


FIGURA 11.95

2. Mostrar cómo habría que programar la matriz tipo PAL de la Figura 11.96 para implementar cada una de las siguientes expresiones suma de productos. Se utiliza una X para indicar un enlace conectado. Simplificar las expresiones si fuera necesario, para que la expresión quepa en la matriz de tipo PAL mostrada.

(a) $Y = A\bar{B}C + \bar{A}B\bar{C} + ABC$

(b) $Y = A\bar{B}C + \bar{A}BC + AB\bar{C} + \bar{A}BC$

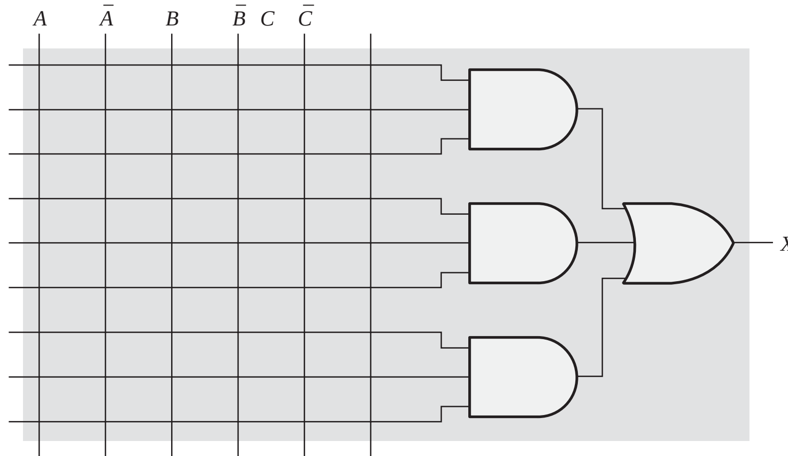


FIGURA 11.96

3. Interpretar cada uno de los siguientes números de dispositivo PAL
   (a) PAL16L2
   (b) PAL12H6
4. Explicar cómo funciona una salida de polaridad programable en una PAL.
5. Indicar en qué se diferencia un dispositivo CPLD de un SPLD.

SECCIÓN 11.2 Dispositivos CPLD de Altera

6. Utilizando el diagrama de bloques de los dispositivos MAX 7000 de la Figura 11.11, determinar el número de:
   (a) entradas de la PIA a un bloque LAB
   (b) salidas de un bloque LAB a la PIA
   (c) entradas desde un bloque de control de E/S hacia la PIA
   (d) salida de un bloque LAB hacia un bloque de control de E/S
7. Determinar el término producto para la puerta AND de una matriz CPLD que se muestra en la Figura 11.97(a). Si se expande la puerta AND, como se muestra en la Figura 11.97(b), determinar la salida suma de productos.

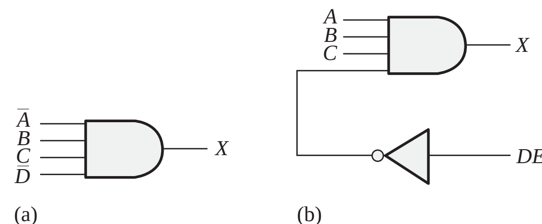


FIGURA 11.97

8. Determinar la salida de la lógica de macrocelda que se muestra en la Figura 11.98 si se aplica $AB\bar{C}D + \bar{A}BCD$ a la entrada de términos de expansión paralelo.

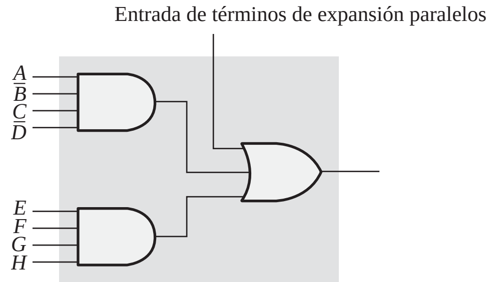


FIGURA 11.98

SECCIÓN 11.3 Dispositivos CPLD de Xilinx

9. Determinar la salida de la PLA de la Figura 11.99. Las X representan enlaces conectados.

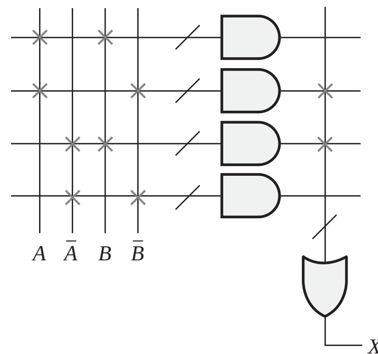


FIGURA 11.99

10. Utilizando el diagrama de bloques del dispositivo CPLD CoolRunner II de la Figura 11.21, determinar el número de:

(a) entradas de la AIM a un FB **(b)** salidas de un FB hacia la AIM

(c) entradas de un bloque de E/S hacia la AIM **(d)** salidas de un FB a un bloque de E/S

11. Determinar las expresiones de salida para X_1 y X_2 en las macroceldas 1 y 2 de la Figura 11.100.

SECCIÓN 11.4 Macroceldas

12. Determinar los datos de salida para el multiplexor de la Figura 11.101, en cada una de las condiciones siguientes:

(a) $D_0 = 1$, $D_1 = 0$, Selección = 0 **(b)** $D_0 = 1$, $D_1 = 0$, Selección = 1

13. Determinar cómo está configurada (combinacional o registrada) la macrocelda de la Figura 11.102 y el bit de datos que hay a la salida (hacia la E/S) para cada una de las siguientes condiciones. El flip-flop es de tipo D. Utilizar la Figura 11.101 para ver la disposición de las entradas de datos de los multiplexores.

(a) Salida XOR = 1, salida del flip-flop $Q = 1$, de entrada E/S = 1, selección de MUX 1 = 1, selección de MUX 2 = 0, selección de MUX 3 = 0, selección de MUX 4 = 0 y selección de MUX 5 = 0.

(b) Salida XOR = 0, salida del flip-flop $Q = 0$, de entrada E/S = 1, selección de MUX 1 = 1, selección de MUX 2 = 0, selección de MUX 3 = 1, selección de MUX 4 = 0 y selección de MUX 5 = 1.

14. Para la macrocelda CPLD de la Figura 11.103, se programan las siguientes condiciones: selección de MUX 1 = 1, selección de MUX 2 = 1, entradas de selección de MUX 3 = 01, selección de MUX 4 = 0, selección de MUX 5 = 1, entradas de selección de MUX 6 = 11, entradas de selección de MUX 7 = 11, selección de MUX 8 =1 y la salida de la puerta OR = 1. El

flip-flop es de tipo D y las entradas de los multiplexores están ordenadas con D_0 en la parte superior y D_n en la parte inferior.

(a) ¿Está la macrocelda configurada para proporcionar lógica combinacional o lógica registrada?

(b) ¿Qué reloj se aplica al flip-flop?

(c) ¿Cuál es el bit de datos en la entrada *D* del flip-flop?

(d) ¿Cuál es la salida de MUX 8?

15. Repetir el Problema 14 para selección de MUX 1 = 0.

SECCIÓN 11.5 Lógica programable: dispositivos FPGA

16. Generalmente, ¿qué elementos componen un bloque lógico configurable (CLB) en una FPGA? ¿Qué elementos componen un módulo lógico?

17. Determinar la expresión de salida de la LUT para las condiciones internas mostrada en la Figura 11.104.

18. Mostrar cómo reprogramar la LUT de la Figura 11.104 para generar la siguiente salida suma de productos:

$$\overline{A}B\overline{C} + A\overline{B}\overline{C} + ABC$$

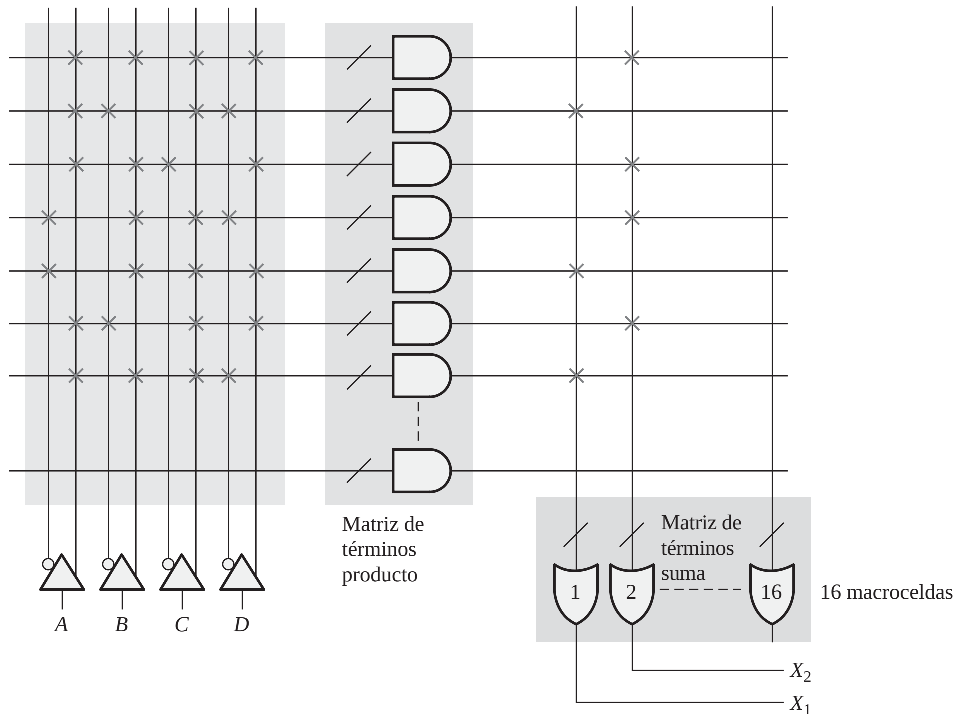


FIGURA 11.100

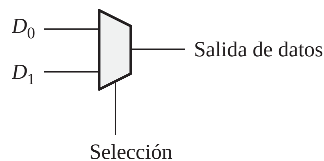


FIGURA 11.101

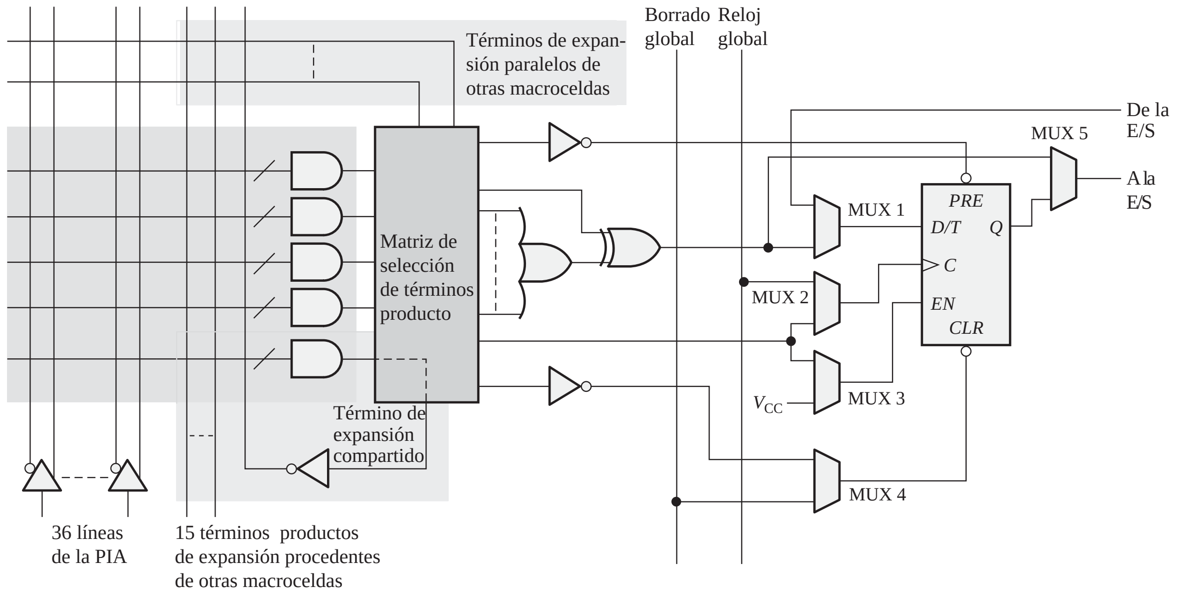


FIGURA 11.102

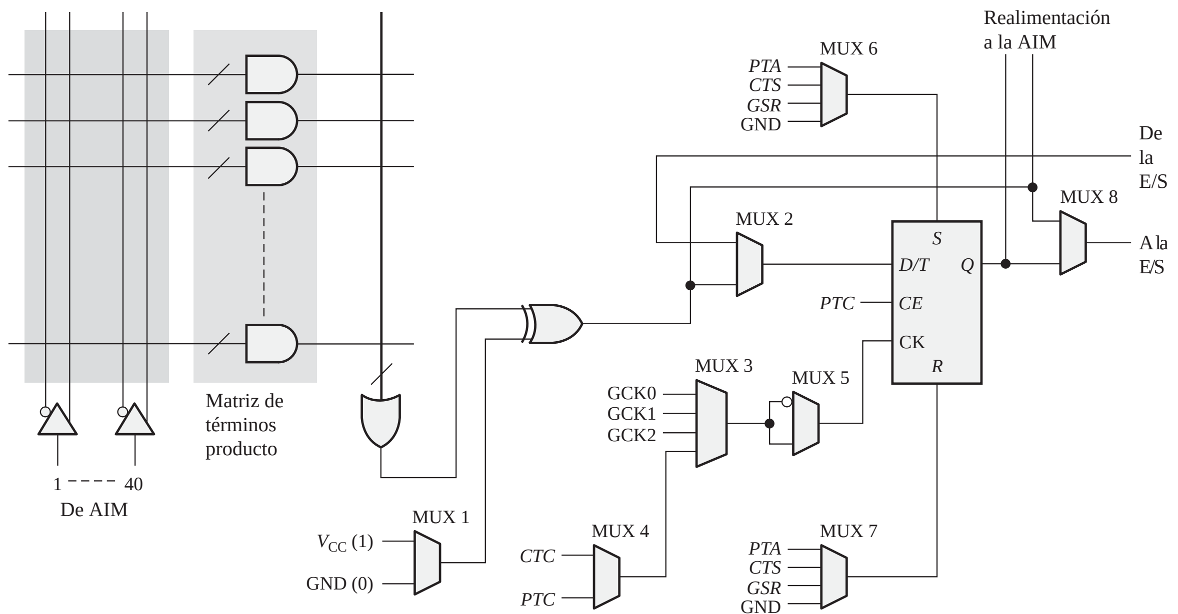


FIGURA 11.103

SECCIÓN 11.6 Dispositivos FPGA de Altera

19. Nombrar los elementos básicos que forman un módulo lógico adaptativo (ALM) en la FPGA Stratix II.

20. Enumerar los modos de operación de un ALM.

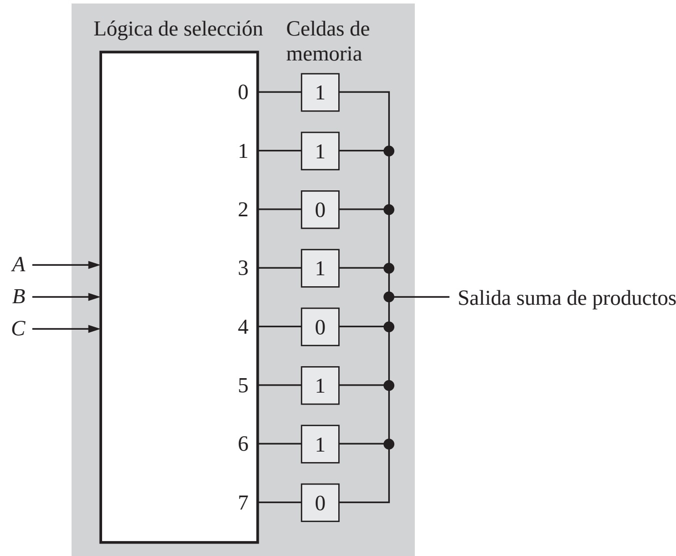


FIGURA 11.104

21. Mostrar un ALM configurado en el modo normal para generar una función suma de productos de 4 variables y una función suma de productos de 2-variables.

22. Determinar la función de salida suma de productos final para el ALM mostrado en la Figura 11.105.

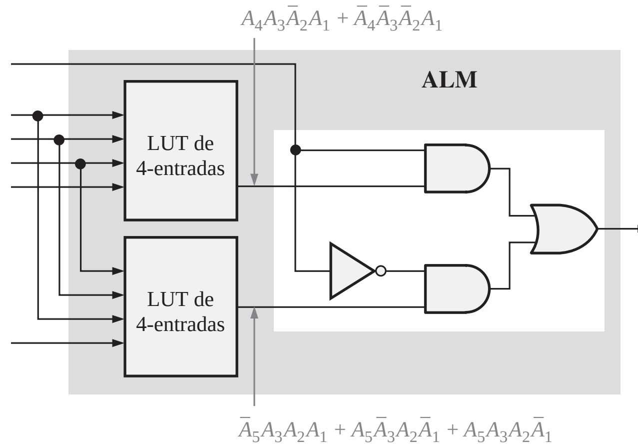


FIGURA 11.105

SECCIÓN 11.7 Dispositivos FPGA de Xilinx

23. Utilizar una o más de las *slices* de la Figura 11.106 para generar la función suma de productos:

$$A_7A_6A_5A_4A_3A_2A_1A_0 + B_7B_6B_5B_4B_3B_2B_1B_0$$

24. En la Figura 11.106 se muestra una *slice* de una FPGA Virtex. Mostrar cómo pueden configurarse una o más de estas *slices* para generar la función suma de productos:

$$A_7\bar{A}_6A_5A_4 + \bar{A}_3A_2A_1\bar{A}_0 + \bar{B}_7\bar{B}_6B_5B_4 + B_3\bar{B}_2\bar{B}_1B_0$$

Suponer que los elementos en gris más claro asi como las tablas LUT son reconfigurables.

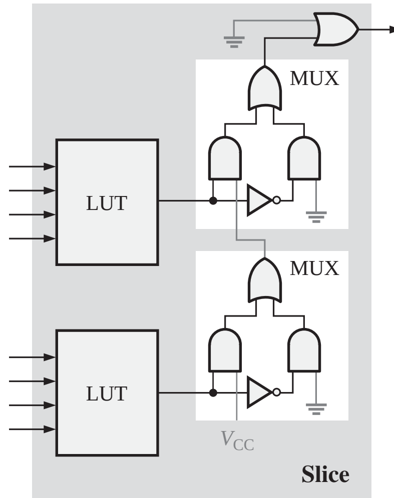


FIGURA 11.106

25. Determinar el número de *slices* (Figura 11.106) necesarias para generar la expresión:

$A_7A_6A_5A_4A_3A_2A_1A_0$

26. Determinar el número de *slices* necesarias para generar la expresión:

$$A_7A_6A_5A_4A_3A_2A_1A_0 + B_7B_6B_5B_4B_3B_2B_1B_0 + C_7C_6C_5C_4C_3C_2C_1C_0$$

SECCIÓN 11.8 Software de lógica programable

27. Mostrar el diagrama lógico que se introduciría en el editor gráfico para definir el circuito descrito por cada uno de los siguientes programas VHDL.

a.
```
entity AND_OR is
    port (A0, A1, A2, A3: in bit; X: out bit);
end entity AND_OR;
architecture LogicFunction of AND_OR is
begin
    X <= (A0 and A1) or (A2 and not A3);
end architecture LogicFunction;
```

b.
```
entity LogicCircuit is
    port (A, B, C, D: in bit; X: out bit);
end entity LogicCircuit;
architecture Function of LogicCircuit is
begin
    X <= (A and B) or (C and D) and
            (A and not B) and (not C and not D);
end architecture Function;
```

28. Mostrar el circuito lógico que habría que introducir en el editor gráfico para implementar la siguiente expresión booleana. Si es posible, simplificar la expresión antes de introducirla.

$$X = \overline{A}BCD + A\overline{B}CD + AB\overline{C}D + ABC\overline{D} + ABCD + \overline{A}\overline{B}\overline{C}\overline{D}$$

29. En la Figura 11.107 se muestra el editor de formas de onda con las formas de onda de entrada para el circuito lógico descrito en el Problema 28. Determinar la forma de onda de salida que se produciría después de ejecutar una simulación.

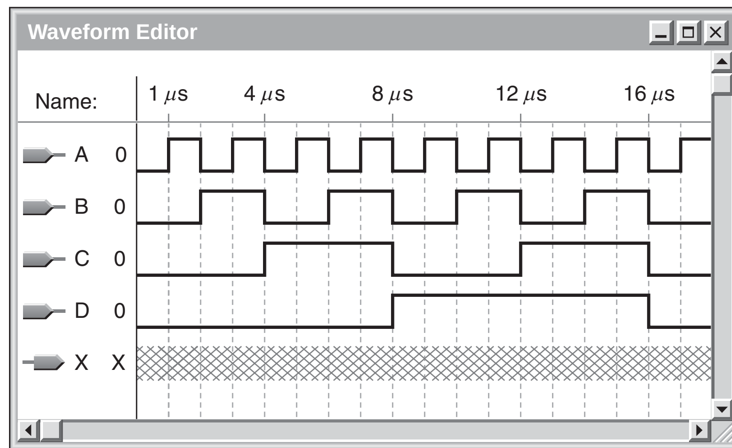


FIGURA 11.107

30. Repetir el Problema 29 para la siguiente expresión booleana:

$$X = \bar{A}BC\bar{D} + A\bar{B}\bar{C}D + ABCD + A\bar{B}C\bar{D} + \bar{A}B\bar{C}D$$

SECCIÓN 11.9 Lógica de exploración de contorno

31. En una determinada celda de exploración de contorno, suponga que los datos fluyen en serie desde la BSC anterior a la BSC siguiente, describir lo que sucede a medida que los datos pasan a través de la BSC indicada.

32. Describir las condiciones y lo que sucede en una BSC determinada cuando los datos fluyen directamente desde la lógica programable interna a un pin de salida del dispositivo.

33. Describir las condiciones y lo que sucede en una BSC determinada cuando los datos fluyen desde un pin de entrada del dispositivo a la lógica programable interna.

34. Describir el camino de los datos para transferir datos desde la línea SDI a la lógica programable interna.

SECCIÓN 11.10 Localización de averías

35. Desarrollar un patrón de bits de prueba para exploración de contorno con el fin de comprobar la lógica programada en el dispositivo mostrado en la Figura 11.108, para todas las posibles combinaciones de entrada.



Aplicación a los sistemas digitales

36. Si se introduce en el editor gráfico la lógica para los siete segmentos mostrada en la Figura 11.79, como un esquemático plano, ¿cuántos elementos pueden eliminarse y qué elementos son esos?

37. En la Figura 11.109 se muestra la pantalla del editor de formas de onda con una simulación para la lógica de 7-segmentos. Determinar cuál puede ser el problema en el circuito simulado.

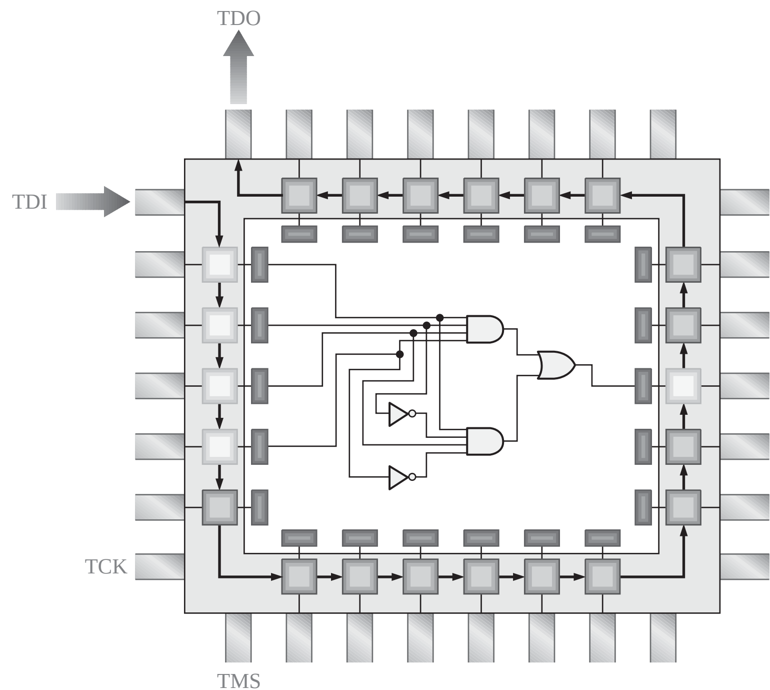


FIGURA 11.108

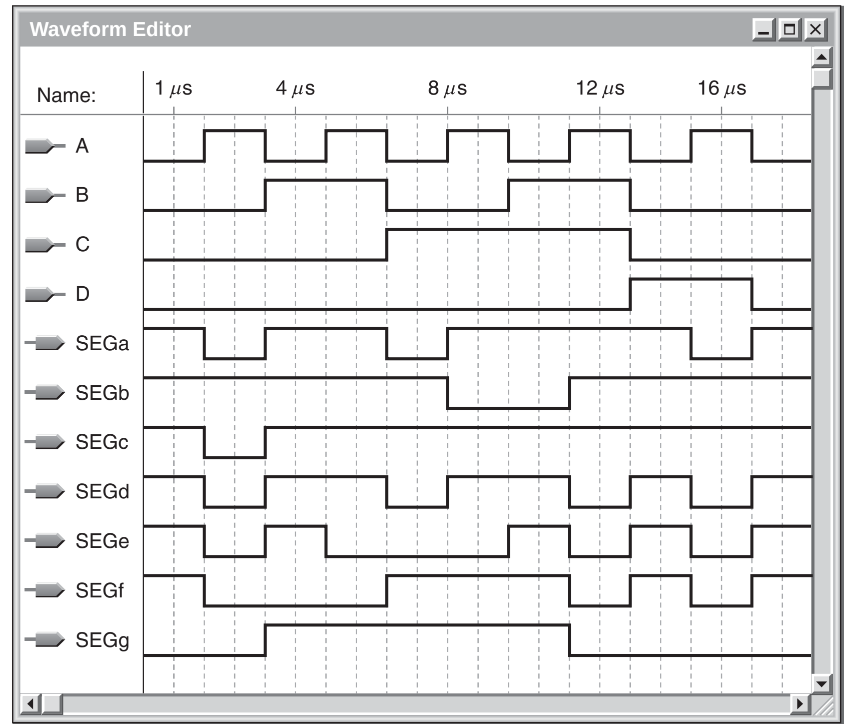


FIGURA 11.109

RESPUESTAS

REVISIONES DE LAS SECCIONES

SECCIÓN 11.1 Lógica programable: SPLD y CPLD

1. PAL: *Programmable Array Logic*, dispositivo lógico de matriz programable.
2. GAL: *Generic Array Logic*, dispositivo lógico de matriz genérica.
3. Una GAL es reprogramable. Una PAL sólo puede programarse una vez.
4. Básicamente, una macrocelda está compuesta de una puerta OR y la lógica de salida asociada, incluyendo un flip-flop.
5. CPLD: *Complex Programmable Logic Device*, dispositivo lógico programable complejo.

SECCIÓN 11.2 Dispositivos CPLD de Altera

1. LAB: *Logic Array Block*, bloque de matriz lógica.
2. En la familia MAX 7000, una LAB está compuesta por 16 macroceldas.
3. Un expansor compartido se emplea para aumentar el número de términos producto en una macrocelda, combinando mediante AND términos suma adicionales (términos producto complementados) procedentes de otras macroceldas.
4. Un expansor paralelo se utiliza para aumentar el número de términos producto de una macrocelda, combinando mediante OR términos producto no utilizados procedentes de otras macroceldas de un bloque LAB.
5. Los dispositivos MAX II están organizados según una arquitectura de fila/columnas y utilizan tablas LUT en sus macroceldas. Los dispsoitivos MAX 7000 están organizados con una arquitectura de columnas tradicional y emplean lógica de suma de productos en sus macroceldas.

SECCIÓN 11.3 Dispositivos CPLD de Xilinx

1. Altera utiliza una arquitectura PAL. Xilinx emplea una arquitectura PLA.
2. Una PLA tiene una matriz AND programable y una matriz OR programable.
3. Una PAL tiene una matriz OR fija.
4. FB: *Function Block,* bloque funcional.

SECCIÓN 11.4 Macroceldas

1. La puerta XOR se utiliza como un inversor programable para los datos. Se puede programar para invertir o no invertir los datos.
2. Combinacional y registrado.
3. El término registrado hace referencia a la utilización de un flip-flop.
4. Multiplexor.

SECCIÓN 11.5 Lógica programable: dispositivos FPGA

1. Generalmente, una FPGA está organizada con una estructura de interconexión en filas/columnas y utiliza tablas LUT en lugar de lógica AND/OR para generar funciones lógicas combinacionales.
2. CLB: *Configurable Logic Block*, bloque lógico configurable.
3. LUT: *Look-Up Table*, tabla de consulta. Un tipo de memoria programable que se utiliza para almacenar y generar funciones de lógica combinacional.
4. Una interconexión local se emplea para conectar los módulos lógicos dentro de un CLB. Una interconexión global se utiliza para conectar un CLB con otros bloques CLB.
5. Un módulo es una parte de lógica integrada en una FPGA para proporcionar una función específica.

6. El término *propiedad intelectual* hace referencia a los diseños de módulos hardware desarrollados por los fabricantes de dispositivos FPGA y que son propiedad de los mismos.

SECCIÓN 11.6 Dispositivos FPGA de Altera

1. La unidad básica de diseño en los dispositivos Stratix II es el bloque LAB.
2. Típicamente, hay ocho módulos ALM en un bloque LAB.
3. Una tabla LUT genera funciones de lógica combinacional en un módulo ALM.
4. Dos.
5. Memoria y DSP (*Digital Signal Processing,* procesamiento digital de la señal).

SECCIÓN 11.7 Dispositivos FPGA de Xilinx

1. Un bloque CLB consta de ocho celdas lógicas o cuatro *slices.*
2. Una celda lógica (LC, *Logic Cell*) está compuesta por una tabla LUT y su lógica asociada.
3. Un *slice* está compuesta por dos celdas lógicas (LC).
4. Una cadena de conexión en cascada está formada por dos o más *slices* conectados para expandir una expresión suma de productos.
5. ASMBL: *Application Specific Modular Block* (bloque modular específico de la aplicación).

SECCIÓN 11.8 Software de lógica programable

1. Introducción del diseño, simulación funcional, síntesis, implementación, simulación de temporización, descarga.
2. Una computadora donde se ejecuta el software de desarrollo para dispositivos PLD, una utilidad de programación o una tarjeta de desarrollo, y un cable de interfaz.
3. Una lista de interconexiones proporciona la información necesaria para describir un circuito.
4. La simulación funcional se realiza antes de la simulación de temporización.

SECCIÓN 11.9 Lógica de exploración de contorno

1. TDI, TMS, TCK, TRST, TDO.
2. TAP: *Test Access Port*, puerto de acceso de prueba.
3. Registro de exploración de contorno, registro de puenteo, registro de instrucción y puerto TAP.
4. Tranferencia de datos de SDI a SDO, transferencia de datos desde la lógica programable interna a un pin de salida del dispositivo, transferencia de datos desde un pin de entrada del dispositivo a la lógica programable interna, transferencia de datos de SDI a la lógica programable interna y transferencia de datos desde SDI al pin de salida del dispositivo.

SECCIÓN 11.10 Localización de averías

1. La prueba mediante cama de pinchos utiliza un aparejo compuesto por una matriz fija de sondas de prueba (que parecen pinchos) sobre la que se coloca la tarjeta de circuito que haya que probar. Cada sonda de prueba hace contacto con un punto de prueba de la tarjeta de circuito, de modo que se pueden realizar las medidas necesarias.
2. El método de la cama de pinchos está limitado por la densidad de los dispositivos lógicos programables, que hace que muchos contactos de un dispositivo sean inaccesibles.
3. En un equipo de sonda volante, una o más sondas se mueven de un punto de prueba a otro sobre una tarjeta de circuito siguiendo un patrón controlado.
4. La exploración de contorno permite la realización de pruebas internas y la programación de un dispositivo lógico programable, asi como la prueba de las interconexiones entre dos o más dispositivos. Este método está basado en el estándar JTAG IEEE Std. 1149.1. La técnica de exploración de contorno utiliza para las pruebas una lógica interna específica del dispositivo.

5. *Intest* y *Extest*.
6. TDI, TDO, TCK, TMS.
7. BSDL: *Boundary Scan Description Language*, lenguaje de descripción para exploración de contorno.

PROBLEMAS RELACIONADOS

11.1. $X = \overline{B}C + \overline{A}B\overline{C} + \overline{A}\overline{B} + C$

11.2 Dieciseis.

11.3 Dieciseis; dieciseis.

11.4 *Véase* la Figura 11.110.

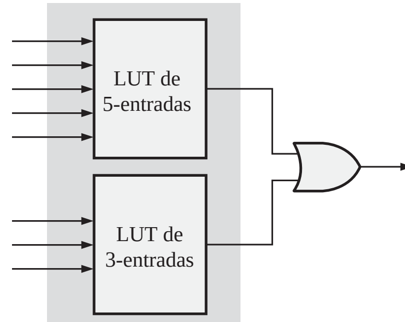


FIGURA 11.110

11.5 *Véase* la Figura 11.111.

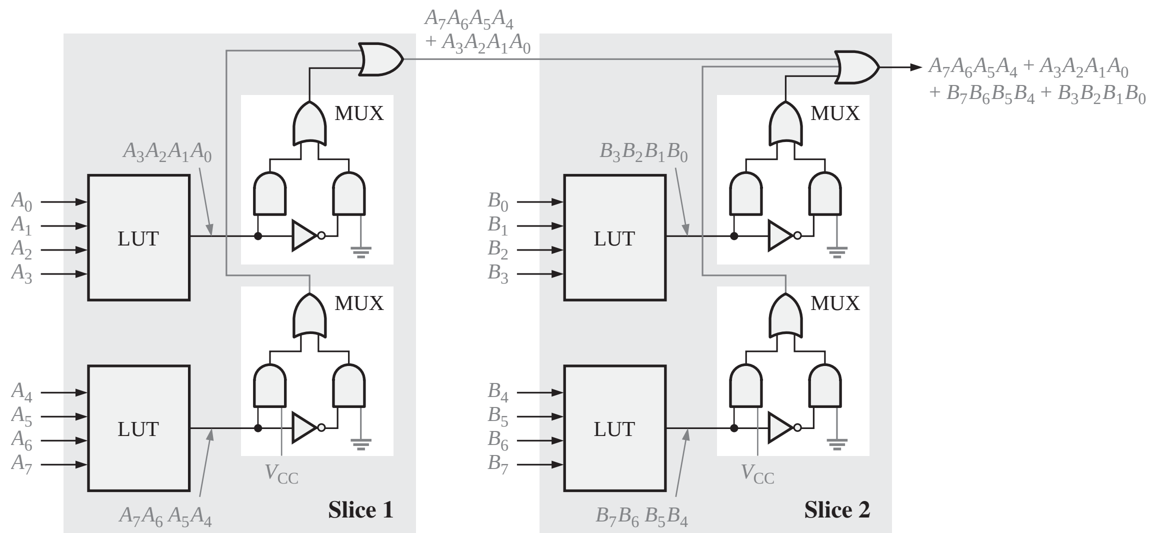


FIGURA 11.111

AUTOTEST

1. (c) **2.** (b) **3.** (a) **4.** (c) **5.** (b) **6.** (b) **7.** (a) **8.** (a) **9.** (c)
10. (a) **11.** (b) **12.** (c) **13.** (b) **14.** (b) **15.** (a) **16.** (a) **17.** (c) **18.** (a)
19. (d) **20.** (c) **21.** (b) **22.** (c) **23.** (b) **24.** (b) **25.** (d)

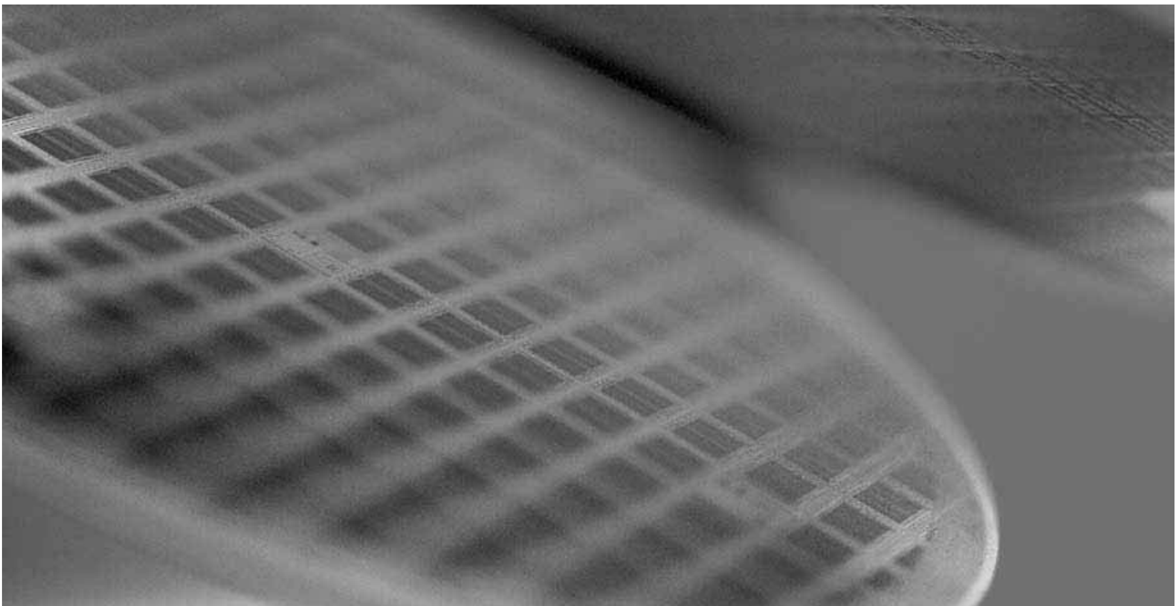
12

INTRODUCCIÓN A LAS COMPUTADORAS

CONTENIDO DEL CAPÍTULO

OBJETIVOS DEL CAPÍTULO

- Nombrar las unidades básicas de una computadora.
- Indicar los elementos básicos de un microprocesador.
- Explicar el funcionamiento básico de una CPU de Intel
- Explicar la arquitectura básica del microprocesador Intel.
- Explicar la operación de bus multiplexada del microprocesador Intel.
- Analizar el modelo software de los procesadores Pentium de Intel.



- Describir un programa de lenguaje ensamblador simple.
- Describir los siete grupos de instrucciones para el procesador Intel.
- Diferenciar entre lenguaje ensamblador y lenguaje máquina.
- Comparar los mecanismos de E/S con sondeo, E/S dirigida por interrupciones e interrupciones software.
- Describir las funciones de los dispositivos PIC y PPI.
- Definir y explicar las ventajas del mecanismo DMA.
- Explicar la comunicación con las distintas funciones mediante el uso de los sistemas de bus.
- Definir las características básicas y aplicaciones de los estándares PCI e ISA de bus interno.
- Definir las características básicas y aplicaciones de los estándares RS-232C, IEEE 1394 (FireWire), USB, IEEE 488 (GPIB) y SCSI de bus externo.

PALABRAS CLAVE

- Puerto
- Programa
- CPU
- Interrupciones
- Periféricos
- Microprocesador
- Bus de direcciones
- Bus de datos
- Bus de control
- Lenguaje máquina
- Lenguaje ensamblador
- Lenguaje de alto nivel
- Triestado
- Módem
- FireWire
- USB
- GPIB
- SCSI

INTRODUCCIÓN

Este capítulo proporciona una breve introducción a las computadoras, microprocesadores y buses. Naturalmente, el tratamiento que se hace en este único capítulo debe forzosamente estar limitado, porque podrían fácilmente dedicarse uno o más capítulos a cada uno de los temas cubiertos en las distintas secciones. Sin embargo, recuerde que nuestro propósito es simplemente proporcionar una introducción básica a estos temas. El tratamiento completo del tema de las computadoras y los microprocesadores se realiza normalmente en un curso posterior. Para obtener más información acerca de los microprocesadores, incluyendo las hojas de características, visite el sitio web de Intel en www.intel.com.

Analizaremos brevemente las familias de microprocesadores de Intel. Utilizaremos el procesador 8086/8088 de Intel como "modelo" para ilustrar los conceptos básicos sobre microprocesadores, describiendo con algo más de detalle las mejoras posteriores experimentadas hasta el desarrollo de la familia Pentium. El 8086/8088 fue la primera generación de la familia 80X86. Aunque el Pentium es más potente y posee numerosas características avanzadas, está relacionado con las familias anteriores tanto en lo que respecta a la arquitectura cuanto en lo que se refiere a las funciones básicas, como por ejemplo la estructura de registros.

12.1 UNA COMPUTADORA BÁSICA

Hay computadoras de propósito especial que permiten controlar diversas funciones en los automóviles o electrodomésticos, controlar procesos de fabricación en las fábricas, proporcionar juegos con propósitos de entretenimiento o actuar como control de sistemas de navegación tales como GPS (*Global Position System*, sistema de posicionamiento global) por nombrar sólo unas cuantas áreas. Sin embargo, el tipo de computadora más familiar es la computadora de propósito general que puede programarse para realizar numerosas tareas distintas.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Describir los elementos básicos de una computadora. ■ Explicar lo que hace cada parte de una computadora. ■ Explicar qué es un dispositivo periférico.

Todas las computadoras están compuestas por bloques funcionales básicos que incluyen una *unidad central de proceso* (CPU, *Central Processing Unit*), una *memoria* y *puertos de entrada/salida.* Estos bloques funcionales se conectan entre sí mediante tres buses internos, como se muestra en el diagrama de bloques de la Figura 12.1. Los tres buses son el *bus de datos*, el *bus de direcciones* y el *bus de control.* Los dispositivos de entrada y salida se conectan a través de los puertos de entrada/salida. Un ***puerto*** es una interfaz física de una computadora, a través de la cual pasan los datos hacia y desde los periféricos.

Las instrucciones y los datos se almacenan en memoria en ubicaciones específicas determinadas por el ***programa***, que es una lista de instrucciones diseñada para resolver un problema específico. Cada ubicación

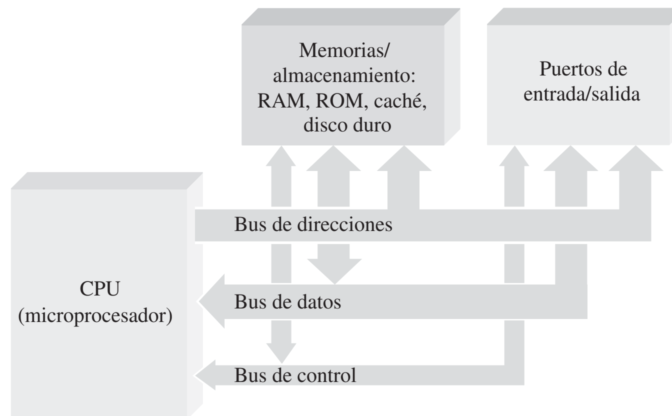


FIGURA 12.1 Diagrama de bloques básico de una computadora.



NOTAS INFORMÁTICAS

Grace Hopper, matemática y una de las primeras programadoras, adquirió unas considerables habilidades de diagnóstico durante su trabajo como oficial de La Marina, en el que utilizaba la computadora Mark I de Harvard, en la década de 1940. Fue ella la que descubrió y documentó el primer fallo real de una computadora, anotándolo en el registro de actividades del Mark I. El fallo había sido debido a una polilla que había quedado atrapada en uno de los relés electromecánicos de la máquina, haciendo que la computadora operara incorrectamente. En inglés, la palabra *bug* sirve para designar a los insectos, por lo que a partir de entonces, cuando alguien preguntaba si se estaban haciendo progresos, los que trabajan con la computadora respondían que estaban "depurando" (*debugging*, que literalmente se podría traducir por desinsectar) el sistema. El término hizo fortuna y la localización de problemas en una computadora (u otro dispositivo electrónico) y en particular en el software se denomina en inglés con el término "*debugging*".

tiene una dirección unívoca asociada. La CPU obtiene las instrucciones colocando una dirección en el bus de direcciones. Las instrucciones se transfieren a través del bus de datos a medida que la CPU las solicita, la CPU ejecuta estas instrucciones de manera secuencial. Frecuentemente, las instrucciones modifican los datos almacenados en la memoria u obtenidos desde un dispositivo de entrada. Los datos procesados pueden volver a almacenarse en memoria o enviarse hacia un dispositivo de salida a través del bus de datos. La CPU genera una serie de señales en el bus de control para coordinar todas estas operaciones.

La CPU

La ***CPU*** (*Central Processing Unit,* unidad central de proceso o UCP) es el "cerebro" de la computadora y se encarga de controlar todo lo que la computadora hace. La CPU es un microprocesador con una serie de circuitos asociados que controla los programas software de la computadora. Básicamente, la CPU obtiene (extrae) cada instrucción de programa de la memoria y lleva a cabo (ejecuta) dicha instrucción.

Después de completar una instrucción, la CPU pasa a la siguiente y en la mayoría de los casos, puede operar con más de una instrucción al mismo tiempo. Este proceso de "extracción y ejecución" se repite hasta que se han ejecutado todas las instrucciones de un programa específico. Por ejemplo, un programa de aplicación puede requerir que se sumen una serie de números. Las instrucciones para sumar los números estarán almacenadas en forma de códigos binarios que indican a la CPU que debe extraer una serie de números de la memoria, sumarlos y volver a almacenar el resultado en la memoria.

Memorias y almacenamiento

En una computadora típica se utilizan diversos tipos de memorias. La RAM (*Random-Access Memory*, memoria de acceso aleatorio) almacena temporalmente datos binarios y programas durante el procesamiento. Los datos son números y otros tipos de información mientras que los programas son listas de instrucciones. En una memoria RAM, pueden leerse y escribirse datos en cualquier momento. La RAM es volátil, lo que quiere decir que la información se pierde si se desconecta la alimentación o si ésta falla. Por tanto, los datos y programas que tengan que ser guardados tienen que transferirse a una memoria no volátil (como por ejemplo un CD o un disco duro) antes de eliminar la alimentación.

La memoria ROM (*Read-Only Memory*, memoria de sólo lectura) almacena un programa permanente del sistema denominado **BIOS** (*Basic Input/Output System*, sistema básico de entrada/salida) y también la información referente a ciertas ubicaciones de los programas del sistema dentro de la memoria. La ROM es no volátil, lo que significa que mantiene la información que se almacene en ella, incluso después de desconectar la alimentación. Como su propio nombre indica, los programas y datos almacenados en ROM no pueden modificarse. En algunas ocasiones, a la memoria ROM se la denomina "*firmware*", porque es un software que tiene carácter permanente dentro de un determinado sistema.

El sistema BIOS forma el nivel más bajo del sistema operativo de una computadora. Contiene instrucciones que indican a la CPU lo que debe hacer en el momento de aplicar la alimentación; la primera instrucción ejecutada se encuentra en el BIOS. Esta memoria especial controla las funciones básicas de arranque de la computadora, que incluyen un autotest y un cargador de arranque de disco con el fin de cargar el resto del sistema operativo. Además, el sistema BIOS almacena las ubicaciones correspondientes a los programas del sistema que gestionan determinadas solicitudes procedentes de los periféricos, denominadas ***interrupciones***, que hace que se detenga automáticamente el procesamiento actual.

La memoria *caché* es una pequeña memoria RAM que se utiliza para almacenar una cantidad limitada de datos frecuentemente utilizados, a los que se puede acceder mucho más rápido que si estuvieran en la memoria RAM principal. La caché almacena información que debe "estar a mano" con el fin de poder volver a utilizarla rápidamente, sin tener que extraerla de nuevo de la memoria principal. La mayoría de los microprocesadores tienen una memoria caché interna denominada de nivel 1 o simplemente L1 (*level* 1). La memoria caché externa se encuentra en un chip de memoria separado, y se la denomina memoria caché de nivel 2 o L2.

El *disco duro* es el principal medio de almacenamiento de una computadora, porque permite almacenar grandes cantidades de datos y es de carácter no volátil. Los niveles más altos del sistema operativo están almacenados en el disco duro al igual que lo están el software de aplicación y los archivos de datos.

El *almacenamiento extraible* forma parte de la mayoría de los sistemas informáticos. Los tipos más comunes de medios de almacenamiento extraibles son los discos CD, los disquetes y los discos Zip (medios de almacenamiento magnético). Los disquetes tienen una capacidad de almacenamiento limitada de unos 1,4 MB (megabyte). Los CD pueden estar disponibles en forma de CD-ROM (*Compact Disk-Read Only Memory*, ROM de disco compacto) y CD-RW (*Rewritable*, reescribible) y pueden almacenar enormes cantidades de datos (normalmente, 650 MB). Los discos Zip almacenan aproximadamente 250 MB.

Puertos de entrada/salida

Generalmente, la computadora envía los datos a un dispositivo periférico a través de un puerto de salida y recibe la información a través de un puerto de entrada. Los puertos pueden configurarse por software con el fin de actuar como puerto de entrada o de salida. El teclado, el ratón, el monitor de vídeo, la impresora y otros periféricos se comunican con la CPU a través de puertos individuales. Generalmente, los puertos se clasifican como puertos serie, que tienen una única línea de datos, o puertos paralelo, que tienen múltiples líneas de datos.

Buses

Los periféricos se conectan a los puertos de la computadora mediante buses de interfaz estándar. Podemos pensar en un bus como en una especie de camino para las señales digitales que está compuesto de un conjunto de conexiones físicas y que posee una serie de especificaciones eléctricas relativas a esas señales. Como ejemplos de buses serie podemos citar *FireWire* y USB (*Universal Serial Bus*, bus serie universal). El bus paralelo más común se denomina simplemente *bus paralelo* y se conecta a un puerto que normalmente se denomina puerto de impresora (aunque este puerto también puede utilizarse para otros periféricos). Otro ejemplo de bus paralelo, que sirve para conectar instrumentos de laboratorio a una computadora, es el GPIB (*General Purpose Interface Bus*, bus de interfaz de propósito general).

Los tres tipos básicos de buses internos que interconectan la CPU con la memoria y el almacenamiento, así como con los puertos de entrada y salida, son el bus de direcciones, el bus de datos y el bus de control. Estos buses se suelen agrupar en lo que se denomina *bus local*. La CPU utiliza el bus de direcciones para especificar posiciones de memoria o direcciones y para seleccionar puertos. El bus de datos se emplea para transferir instrucciones de programa y datos entre la CPU, las memorias y los puertos. El bus de control se utiliza para transferir señales de control hacia y desde la CPU.

Software

Además del hardware, existe otra faceta de gran importancia en una computadora: el software. Es el software el que hace posible que el hardware realice una tarea útil. Las dos categorías principales de software usado en una computadora son el software del sistema y el software de aplicación.

Software del sistema. El software del sistema es el sistema operativo de una computadora y permite al usuario comunicarse con la máquina. Los sistemas operativos más comunes utilizados en equipos de sobremesa y portátiles son Windows, MacOS y UNIX. También existen muchos otros sistemas operativos que se emplean en computadoras de propósito especial o de tipo *mainframe*.

El software del sistema realiza dos funciones básicas. Se encarga de gestionar todo el hardware y el software de aplicación dentro de una computadora. Por ejemplo, el sistema operativo gestiona y asigna el espacio existente en el disco duro. También proporciona una interfaz coherente entre el software de aplicación y

el hardware. Esto permite que un mismo programa de aplicación funcione sobre computadoras distintas que pueden diferir en cuanto a sus detalles hardware.

El sistema operativo de la computadora permite tener varios programas ejecutándose a un mismo tiempo. Esta característica se denomina multitarea. Por ejemplo, podemos estar utilizando un procesador de textos al mismo tiempo que descargamos algo de Internet e imprimimos un mensaje de correo electrónico.

Software de aplicación. El software de aplicación se utiliza para realizar una tarea o trabajo específico. La Tabla 12.1 enumera varios tipos de software de aplicación.

Secuencia de operación. Cuando se enciende por primera vez la computadora, esto es lo que sucede:

1. Se carga en RAM el sistema BIOS de la ROM y se realiza un autotest para comprobar todos los componentes principales y la memoria. Asimismo, el sistema BIOS proporciona información acerca del almacenamiento, la secuencia de arranque y otros tipos de informaciones similares.
2. El sistema operativo (como por ejemplo Windows) situado en el disco duro se carga en la memoria RAM.
3. Los programas de aplicación (como por ejemplo Microsoft Word) están almacenados en el disco duro. Cuando se selecciona uno, se carga en la memoria RAM. En ocasiones, sólo se cargan partes de los programas, según van siendo necesarias.
4. Los archivos requeridos por las aplicaciones se cargan desde el disco duro en la memoria RAM.
5. Cuando un archivo se guarda y la aplicación se cierra, el archivo se vuelve a escribir en el disco y tanto ese archivo como la aplicación se eliminan de la RAM.

Aplicación	Función	Ejemplos
Procesador de textos	Preparación de documentos de texto y cartas	Microsoft Word, WordPerfect
Dibujo	Preparación de dibujos técnicos e imágenes	CorelDraw, Freehand,Illustrator
Hoja de cálculo	Manipulación de números y palabras dispuestas en forma de matriz	Excel, Lotus 123
Autoedición	Preparación de boletines, folletos, libros y otros materiales impresos	Quark XPress, Pagemaker
Fotografía	Manipulación de imágenes digitales y adición de efectos especiales a las imágenes	Photoshop, Image Expert
Contabilidad	Preparación de impuestos, apuntes contables	Quickbooks, Turbotax, MYOB
Presentaciones	Preparación de diapositivas y presentaciones técnicas	PowerPoint, Harvard Graphics
Gestión de datos	Manipulación de grandes bases de datos	Filemaker, Access
Multimedia	Edición de vídeo digital, producción de imágenes en movimiento para presentaciones	Premier, Dreamweaver, After Effects
Reconocimiento de voz	Conversión de voz a texto	NaturallySpeaking
Preparación de sitios web	Herramientas para crear páginas web y sitios web en Internet	FrontPage, Acrobat
Simulación de circuitos	Creación y prueba circuitos electrónicos	Multisim

TABLA 12.1 Software de aplicación.

El sistema informático

El diagrama de bloques de la Figura 12.2 muestra los principales elementos de un sistema informático típico, así como la forma en que se interconectan. Para que una computadora realice una determinada tarea, deberá comunicarse con el "mundo exterior", intercambiando información con las personas, con los dispositivos sensores o con los dispositivos que haya que controlar de alguna manera. Para ello, existe un teclado para la introducción de datos, un ratón, un monitor, una impresora, un módem y una unidad de CD en los sistemas más básicos. Todos estos elementos se denominan ***periféricos***.

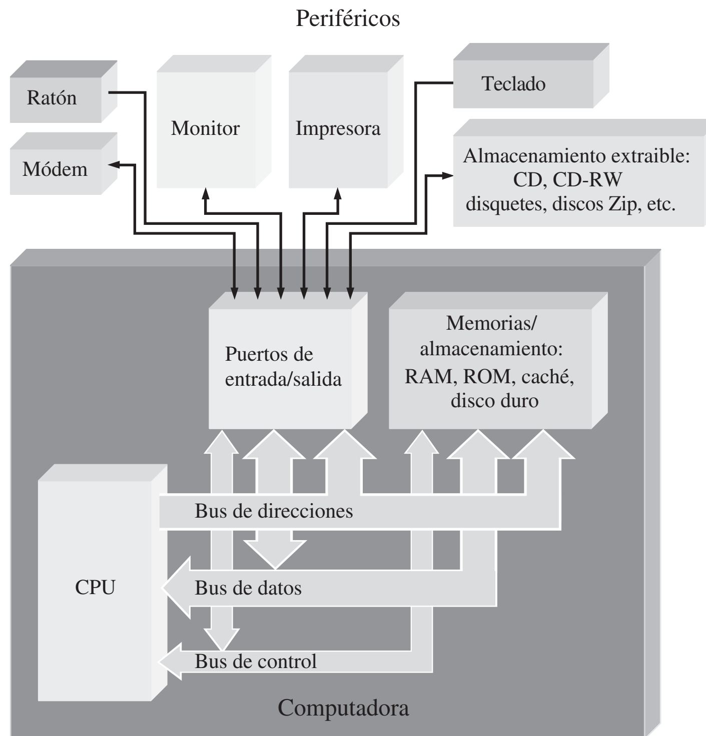


FIGURA 12.2 Diagrama de bloques básico de un sistema informático típico incluyendo algunos periféricos comunes. La propia computadora se muestra dentro del recuadro de color gris más oscuro.

REVISIÓN DE LA SECCIÓN 12.1

Las respuestas se encuentran al final del capítulo

1. ¿Cuáles son los principales elementos o bloques de una computadora?
2. ¿Cuál es la diferencia entre una memoria RAM y una ROM?
3. ¿Qué son los periféricos?
4. ¿Cuál es la diferencia entre el hardware y el software de una computadora?

12.2 MICROPROCESADORES

El ***microprocesador*** es un circuito integrado digital que puede programarse con una serie de instrucciones para que realice diversas operaciones con los datos. Un microprocesador es la CPU de una com-

putadora. Puede llevar a cabo operaciones aritméticas y lógicas, desplazar datos de un lugar a otro y tomar decisiones basándose en ciertas instrucciones.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Describir los elementos básicos de un microprocesador. ■ Explicar los buses de un microprocesador. ■ Describir el conjunto de instrucciones de un microprocesador.

Elementos básicos

Un microprocesador está compuesto por varias unidades, diseñada cada una de ellas para realizar un trabajo específico. Las unidades específicas, junto con su diseño y organización se denominan arquitectura de la computadora. La arquitectura determina el conjunto de instrucciones y el procedimiento que se sigue para ejecutar esas instrucciones. Cuatro unidades básicas que son comunes a todos los microprocesadores son la unidad aritmético lógica (ALU, *Arithmetic Logic Unit*), el decodificador de instrucciones, la matriz de registros y la unidad de control, como se muestra en la Figura 12.3.

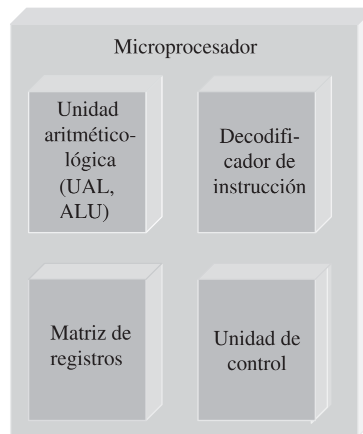


FIGURA 12.3

Unidad aritmético-lógica. La ***ALU*** es el elemento de procesamiento clave del microprocesador. Realiza, dirigida por la unidad de control, operaciones aritméticas (suma, resta, multiplicación y división) y operaciones lógicas (NOT, AD, OR y OR-exclusiva), así como muchos otros tipos de operaciones. Los datos con los que trabaja la ALU se obtienen de la matriz de registros.

Decodificador de instrucciones. El decodificador de instrucciones puede considerarse parte de la ALU, aunque en nuestras explicaciones lo consideraremos una función separada, porque las instrucciones y la decodificación de las mismas resultan cruciales para la operación de un microprocesador. El microprocesador lleva a cabo una determinada tarea bajo control de una serie de programas que están compuestos por listas de instrucciones almacenadas en memoria. El decodificador de instrucciones toma cada instrucción binaria en el orden en el que aparece en la memoria y la decodifica.

Matriz de registros. La **matriz de registros** es una colección de registros contenida en el microprocesador. Durante la ejecución de un programa, los datos y las direcciones de memoria se almacenan temporalmente en los registros que forman esta matriz. La ALU puede acceder a esos registros muy rápidamente, lo que permite que el programa se ejecute de forma más eficiente. Algunos registros se consideran de propósito general, lo que quiere decir que pueden ser empleados para cualquier propósito dictado por el programa. Otros regis-

tros tienen funciones y capacidades específicas y no pueden emplearse como registros de propósito general. Existen, por último, otros registros denominados registros invisibles para el programa, que sólo el microprocesador emplea y que no están disponibles para el programador.

Unidad de control. La **unidad de control** está "a cargo" del procesamiento de las instrucciones una vez que se han decodificado. Proporciona las señales de temporización y de control para introducir y extraer datos en el microprocesador y para sincronizar la ejecución de las instrucciones.

Buses de microprocesador

Los tres buses mencionados anteriormente son las conexiones utilizadas en los microprocesadores para transferir datos, direcciones e instrucciones.

El bus de direcciones. El **bus de direcciones** es una "calle de una sola dirección" a través de la cual el microprocesador envía un código de dirección a la memoria o a otro dispositivo externo. El tamaño o anchura del bus de direcciones está especificado por el número de hilos conductores o pines. Los primeros microprocesadores tenían dieciséis líneas de direcciones que permitían seleccionar 65.536 (2^{16}) posiciones distintas de memoria. Cuantos más bits compongan la dirección, mayor será el número de posiciones de memoria a las que se podrá acceder. El número de bits de direcciones ha aumentado hasta el punto de que un Pentium dispone de 36 bits de direcciones y puede acceder a más de 68 G (68.000.000.000) posiciones de memoria.

El bus de datos. El **bus de datos** es una "calle de dos direcciones" a través de la cual se transfieren datos o códigos de instrucción hacia el microprocesador o se envían hacia el exterior los resultados de las operaciones o cálculos. Los microprocesadores originales tenían buses de datos de 8 bits, mientras que los microprocesadores actuales tienen buses de datos de hasta 64 bits.

El bus de control. El **bus de control** es utilizado por el microprocesador para coordinar sus operaciones y para comunicarse con los dispositivos externos. El bus de control dispone de señales que permiten leer y escribir datos en memoria o realizar una operación de entrada/salida en el instante adecuado. Líneas del bus de control también se usan para insertar estados de espera especiales con el fin de adaptarse a dispositivos más lentos y evitar la contienda de bus, una condición que puede producirse si hay dos o más dispositivos intentando comunicarse a un mismo tiempo.

Programación de un microprocesador

Todos los microprocesadores funcionan con un conjunto de operaciones que implementa las operaciones básicas. Por ejemplo, el Pentium tienen centenares de variantes en su conjunto de instrucciones, pudiendo dividirse dicho conjunto en siete grupos básicos:

- Transferencia de datos
- Aritméticas y lógicas
- Manipulación de bits
- Bucles y saltos
- Manipulación de cadenas
- Subrutinas e interrupciones
- Control

Cada instrucción consta de un grupo de bits (1s y 0s) que el microprocesador decodifica antes de ejecutar la instrucción. Estas instrucciones en código binario se denominan lenguaje máquina y éste es el único tipo de lenguaje que el microprocesador reconoce. Las primeras computadoras se programaban escribiendo esas

instrucciones en código binario, lo que era un trabajo muy tedioso y bastante proclive a errores. Este método primitivo de programación en código binario ha evolucionado a otra forma más elaborada en la que las instrucciones codificadas se representan mediante palabras similares a las del idioma inglés, con el fin de formar lo que se conoce como lenguaje ensamblador. Hablaremos más detalle de este tipo de lenguaje en la Sección 12.4.

Progreso tecnológico

El primer microprocesador, el Intel 4004, fue presentado en 1971. Básicamente, lo único que podía hacer era suma y restar de cuatro en cuatro bits. En 1974, el Intel 8080 se convirtió en el primer microprocesador utilizado como CPU en una computadora. El chip 8080 tenía 6000 transistores, un bus de datos de 8 bits y funciona con una frecuencia de reloj de 2 MHz. El 8080 podía ejecutar unos 0,64 millones de instrucciones por segundo (MIPS, *Million Instructions Per Second*). La familia Intel ha evolucionado desde el 8080, pasando por varias generaciones de procesadores distintos, hasta llegar al Pentium 4. Este último microprocesador (aunque puede que ya no sea el último cuando este libro llegue a manos del lector) contiene unos 42.000.000 de transistores en el chip y un bus de datos de 64 bits. Funciona con frecuencias de reloj de hasta 3 GHz o más y puede ejecutar aproximadamente 1700 MIPS. Los conjuntos de instrucciones también han evolucionado enormemente, pero el Pentium 4 puede ejecutar cualquier código de instrucción que se ejecutara en el 8086, que es el dispositivo que apareció en el mercado en 1979, después del 8080.

El número de transistores disponible tiene un tremendo impacto sobre las prestaciones y el tipo de tareas que el procesador puede llevar a cabo. Por ejemplo, el gran número de transistores que puedan incorporarse en un mismo chip ha hecho que sea posible una tecnología denominada *pipelining*. Básicamente, la tecnología *pipelining* permite que en un momento determinado haya más de una instrucción en proceso de ejecución. Asimismo, los microprocesadores modernos disponen de múltiples decodificadores de instrucciones cada uno con su propia *pipeline*. Esto permite procesar simultáneamente varios flujos de instrucciones.

REVISIÓN DE LA SECCIÓN 12.2

1. ¿Cuáles son los cuatro elementos básicos de un microprocesador?
2. ¿Cuáles son los tres tipos de buses disponibles en un microprocesador?
3. ¿Qué función realiza un microprocesador en una computadora?
4. ¿Cuáles son las tres operaciones básicas que realiza un microprocesador?
5. ¿Qué es el *pipelining*?

12.3 UNA FAMILIA ESPECÍFICA DE MICROPROCESADORES

La familia original de microprocesadores Intel ha experimentado enormes cambios a lo largo de los años, evolucionando desde el 8086/8088 hasta la familia Pentium, y manifestándose esa evolución tanto en términos de velocidad como de complejidad. Sin embargo, el conjunto de registros básico y otras características del 8086/8088 se han conservado (y ampliado) a lo largo de este proceso evolutivo, por lo que todos los procesadores Intel más modernos responden a las mismas instrucciones (además de a una serie de instrucciones nuevas) que los dispositivos originales. En esta sección vamos a comenzar con una breve introducción a los conceptos básicos de la arquitectura, operación y programación de un microprocesador. La sección termina con una panorámica de los cambios principales en la estructura de registros que forma el modelo software de los procesadores más recientes. Nuestro objetivo es presentar un procesador básico y analizar las mejoras experimentadas por la familia Intel a lo largo de su evolución.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Explicar la operación básica de un microprocesador. ■ Describir la unidad de interfaz de bus. ■ Establecer el propósito de los registros de segmentos. ■ Establecer el propósito del puntero de instrucción. ■ Describir la unidad de ejecución. ■ Describir el conjunto general de registros. ■ Establecer el propósito del registro de indicadores. ■ Explicar el modelo software del procesador Pentium.

Operación básica

Un microprocesador ejecuta un programa pasando repetidamente por los tres pasos siguientes:

1. Extraer una instrucción de la memoria y colocarla en la CPU.
2. Decodificar la instrucción; si la instrucción requiere alguna otra información, extraer dicha información. En el paso de decodificación, se actualiza el contador de programa para que apunte a la instrucción siguiente.
3. Ejecutar la instrucción (hacer lo que la instrucción indica). Los resultados se devuelven a los registros y a la memoria en este paso.

La arquitectura del microprocesador 8086/8088 proporcionaba dos unidades internas separadas: la unidad de ejecución (**EU**, *Execution Unit*), que ejecuta las instrucciones y la unidad de interfaz de bus (**BIU**, *Bus Interface Unit*), que implementa la interfaz con los buses del sistema y se encarga de extraer las instrucciones, leer los operandos y escribir los resultados. Estas unidades se muestran en la Figura 12.4.

La BIU realiza todas las operaciones de bus bajo control de la unidad de ejecución (EU), como por ejemplo las transferencias de datos de la memoria o de las operaciones de E/S. Mientras que la EU está ejecutando instrucciones, la BIU "se anticipa" y extrae más instrucciones de la memoria. Esta acción se denomina **precarga** o *pipelining.* El concepto de precarga consiste en permitir al procesador ejecutar instrucciones al mismo tiempo que la siguiente instrucción se está extrayendo, eliminando así los tiempos de inactividad. Las instrucciones precargadas se almacenan en una memoria interna de alta velocidad, denominada **cola** de instrucciones. La cola de instrucciones permite a la BIU mantener el suministro de instrucciones hacia la EU. La EU no tiene que esperar a que se extraiga de memoria la siguiente instrucción; sino que extrae esa instrucción directamente de la cola en mucho menos tiempo. En el Pentium, este proceso va un paso más allá: dos unidades de ejecución completas permiten ejecutar al mismo tiempo dos instrucciones distintas, siempre y cuando sean independientes. Ciertos compiladores están diseñados para aprovecharse de las dos unidades de ejecución, en un proceso conocido como **emparejamiento de instrucciones** mediante el que se eliminan las dependencias.

Arquitectura básica del 8086/8088

La Figura 12.5 muestra un diagrama de bloques de la arquitectura (organización interna) de un microprocesador 8088. Externamente, el 8088 tiene 20 bits de dirección, que permiten direccionar 1 MB (1.048.576 bytes) de memoria y utilizaba un bus de datos de 8 bits. Internamente, el 8088 tenía un bus de datos de 16 bits y una cola de 4 bytes. El 8086 era idéntico salvo porque disponía de un bus de datos externo de 16 bits y una cola de instrucciones de 6 bytes.

La unidad de interfaz de bus (BIU)

Los componentes principales de la BIU son la cola de instrucciones de 4 bytes, los registros de segmentos (CS, DS, SS y ES), el puntero de instrucción (IP) y el bloque de suma de direcciones (Σ). Los buses internos de datos de 16 bits y el bus Q interconectan la BIU y la unidad de ejecución EU.

Cola de instrucciones La cola de instrucciones incrementa la velocidad media de ejecución de los programas (denominada **tasa de transferencia**), almacenando hasta cuatro bytes (seis en el 8086). Como se ha descrito

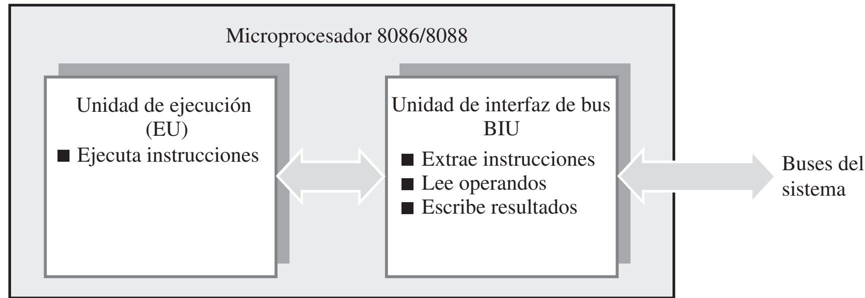


FIGURA 12.4 El 8086/8088 tiene dos unidades internas separadas, la unidad de ejecución (EU) y la unidad de interfaz de bus (BIU).

EU
BIU
Bus de direcciones (20 bits)
Σ
Registros generales
AH AL
BH BL
CH CL
DH DL
SP
BP
DI
SI
Bus de datos (8 bits)
CS
DS
SS
ES
IP
Registros de comunicaciones internas
Lógica de control del bus
Bus 8088
Bus de datos de la ALU (16 bits)
Registros temporales
ALU
Sistema de control de la EU
Bus Q (8 bits)
Cola de instrucciones
1 2 3 4
Indicadores

FIGURA 12.5 Organización interna del microprocesador 8088.

anteriormente, esta técnica permitía al 8088, esencialmente, realizar dos tareas al mismo tiempo: extraer instrucciones y ejecutarlas. Esta funcionalidad se ha ampliado en los procesadores posteriores, para incluir colas mucho más rápidas y de mayor tamaño.

Registros de segmentos Los procesadores 8086/8088 tienen cuatro registros de segmentos (CS, DS, SS y ES) que eran todos ellos registros de 16 bits que se empleaban en el proceso de formación de una dirección de 20 bits. Un **segmento** es un bloque de 64 kB de memoria, que puede dar comienzo en cualquier punto del 1 MB (1.048.576 bytes) del espacio de memoria, siempre y cuando comience en una frontera de 16 bytes (es decir, que sea divisible por 16).

Al diseñar el 8086/8088 y los procesadores subsiguientes, Intel seleccionó un método original para generar la dirección física de 20 bits requerida, utilizando dos registros de 16 bits. Uno de los registros que formaba la dirección física (dirección real) era siempre un registro de segmento; el otro registro era un registro general de 16 bits que contuviera información de dirección. La ventaja principal del método seleccionado era permitir que el código fuese fácilmente reubicable. Un **código reubicable** puede desplazarse a cualquier lugar dentro del espacio de memoria sin modificar el código base.

Cada uno de los cuatro segmentos identifica la dirección de inicio de un bloque de 64 kB (65.536 bytes) que representa una "ventana" del espacio completo de memoria de 1 MB (20 bits). La dirección de inicio de un segmento está representada por el número de 16 bits contenido en el registro de segmento, más cuatro bits implícitos que se añaden a la derecha y que se supone que siempre son cero. En otras palabras, el registro de segmento contiene los 16 bits más significativos de la dirección física de inicio del segmento.

Los cuatro registros de segmento (CS, DS, SS y ES) pueden ser modificados mediante el programa para apuntar a otros bloques de 64 kB según sea necesario. Para fragmentos de código de pequeño tamaño normalmente no es necesario modificar los segmentos. Los cuatro segmentos pueden representar ubicaciones distintas dentro del espacio de memoria, o bien esos segmentos pueden solaparse dependiendo del tamaño y de los requisitos de cada programa concreto. Pueden incluso definirse los segmentos para que apunten al mismo bloque de 64 kB. En el 8086/8088, los segmentos de memoria actualmente direccionables eran los

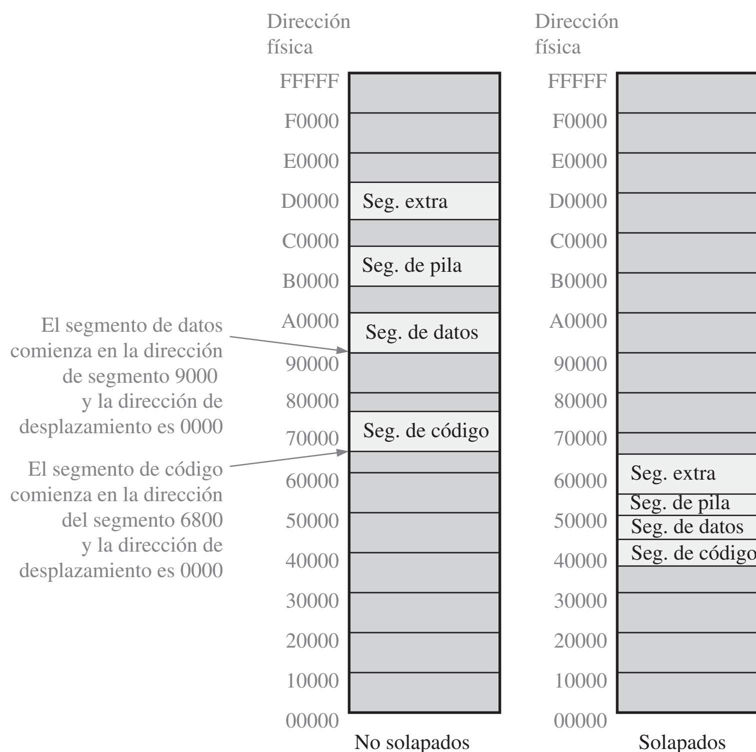


FIGURA 12.6 Segmentos solapados y no solapados dentro del primer MB de memoria. Cada segmento representa 64 kB.

definidos por la dirección de segmento contenida en el registro CS (*Code Segment*, segmento de código), el registro DS (*Data Segment*, segmento de datos), el SS (*Stack Segment*, segmento de pila) y el registro ES (*Extra Segment*, segmento extra). En los procesadores posteriores, se añadieron otros registros de segmento.

Como hemos mencionado, cada segmento es un bloque de 64 kB de memoria. Para localizar una posición de memoria concreta, la dirección de segmento se combina con otra dirección que representa el desplazamiento. La dirección del segmento representa los dieciséis bits más significativos (cuatro dígitos hexadecimales) de la dirección física correspondiente al inicio del segmento. La dirección de desplazamiento son dieciséis bits adicionales que representan la distancia desde el principio del segmento hasta la dirección física deseada dentro del segmento. La Figura 12.6 ilustra la manera en que la memoria se divide en segmentos y muestra ejemplos de segmentos solapados y no solapados.

Puntero de instrucción (IP) y bloque de suma de direcciones El puntero de instrucción **IP** (*Instruction Pointer*) de 16 bits apunta al desplazamiento, dentro de la memoria, de la siguiente instrucción que hay que ejecutar. El IP siempre hace referencia al registro CS (segmento de código); por tanto, la dirección física de la siguiente instrucción se forma combinando el segmento de código y el puntero de instrucción. El IP siempre contiene la dirección de desplazamiento de la siguiente instrucción mientras que el registro CS siempre contiene la dirección del segmento. Esta dirección se muestra en lenguaje ensamblador con el formato CS:IP.

Para formar la dirección física de 20 bits correspondiente a la siguiente instrucción, la dirección de desplazamiento de 16 bits almacenada en el IP se suma a la dirección de segmento contenida en el registro CS, después de desplazar ésta cuatro bits a la izquierda, como se muestra en la Figura 12.7. Como hemos mencionado anteriormente, los cuatro bits menos significativos de la dirección de segmento se consideran siempre 0000. A continuación, la suma se lleva a cabo en el bloque de suma de direcciones.

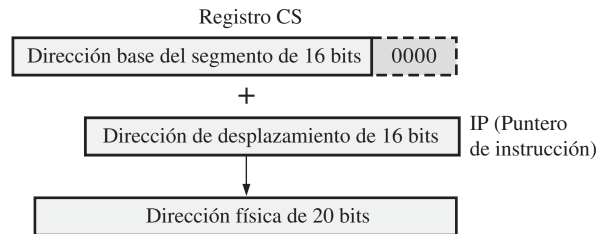


FIGURA 12.7 Formación de la dirección física de 20 bits a partir de la dirección base del segmento y de la dirección de desplazamiento.

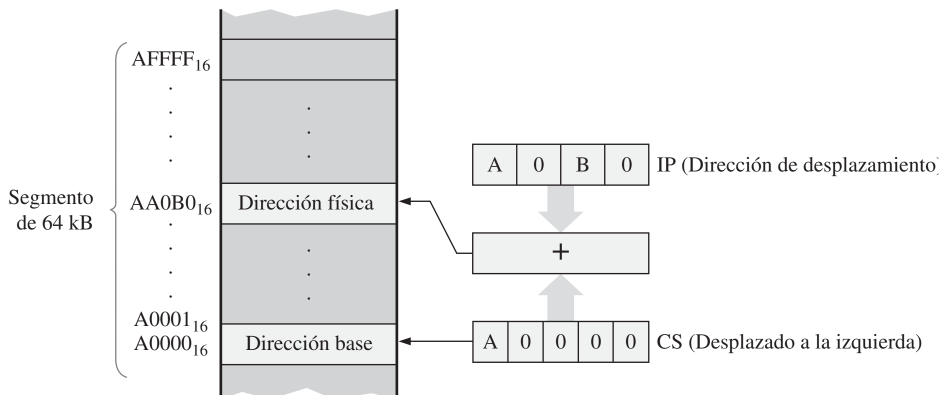


FIGURA 12.8 Ilustración del método de direccionamiento segmentado.

La Figura 12.8 ilustra el direccionamiento de una posición de memoria mediante el método segmento: desplazamiento. En esta figura, $A000_{16}$ es el valor contenido en el registro de segmento y $A0B0_{16}$ es el contenido del puntero de instrucción IP. Cuando desplazamos el registro CS y lo sumamos al registro IP, obtenemos $A0000_{16} + A0B0_{16} = AA0B0_{16}$ como dirección física.

EJEMPLO 12.1

En la Figura 12.9 se muestra el contenido hexadecimal del registro CS y del registro IP. Determinar la dirección física de memoria de la siguiente instrucción.

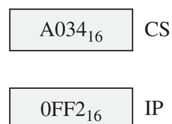


FIGURA 12.9

Solución Desplazando cuatro bits hacia la izquierda la dirección base CS (lo que equivale a desplazar un dígito hexadecimal) colocamos un 0_{16} en la posición menos significativa, como se muestra en la Figura 12.10. La dirección base desplazada y la dirección de desplazamiento se suman para generar la dirección física de 20 bits.

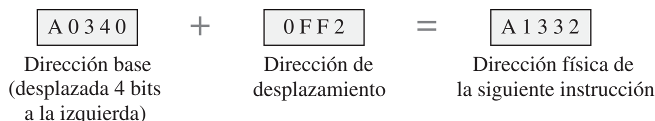


FIGURA 12.10

Problema relacionado* Determinar la dirección física si el contenido del registro CS es $6B4D_{16}$.

* Las respuestas se encuentran al final del capítulo.

Es importante comprender cómo se utiliza el método segmento:desplazamiento para formar la dirección física; sin embargo, durante el trabajo de programación, normalmente no suele ser necesario que el programador especifique las direcciones físicas reales. De esta tarea se encarga el programa ensamblados, utilizando las etiquetas suministradas por el programador. Cuando se necesita una dirección física, el programador la especifica generalmente mediante el método segmento:desplazamiento. Por tanto, la dirección correspondiente al Ejemplo 12.1 podría expresarse simplemente como A034:0FF2.

La unidad de ejecución (EU)

La unidad de ejecución decodifica las instrucciones extraídas por la BIU, genera las apropiadas señales de control y ejecuta las instrucciones. Las partes principales de la EU son la unidad aritmético-lógica (ALU), los registros generales y los indicadores.

La unidad aritmético-lógica Esta unidad realiza todas las operaciones artiméticas y lógicas, trabajando con operandos de 8 o 16 bits.

Los registros generales Este conjunto de registros de 16 bits está dividido en dos conjuntos de cuatro registros cada uno, como se muestra en la Figura 12.11. Uno de los conjuntos está formado por los registros de datos y el otro por los registros de puntero y de índice. Los registros de **puntero** y de índice se emplean generalmente para almacenar desplazamientos (tal como se usa aquí, un puntero hace referencia a una posición de memoria específica). En el caso del puntero de pila (SP, *Stack Pointer*) y del puntero base (BP, *Base Pointer*), la referencia predeterminada para generar una dirección física es el segmento de pila (SS, *Stack Segment*). Los punteros de índice (SI y DI) y el registro base (BX) utilizan generalmente de manera predeterminada el registro de segmento de datos (DS), aunque se hace una excepción a esta regla general para ciertos tipos de instrucciones.

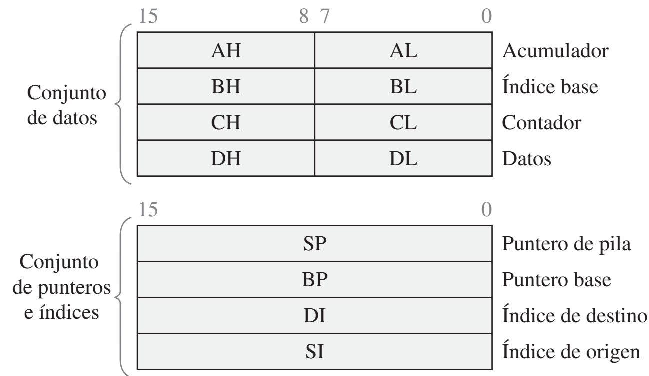


FIGURA 12.11 Conjunto de registros generales.

Cada uno de los registros de datos de 16 bits (AX, BX, CX, DX) tiene dos secciones de 8 bits a las que se puede acceder de manera separada. Dependiendo del programa, pueden utilizarse estos registros como un registro de 16 bits o como dos registros de 8 bits. Los bytes de menor peso de los registros de datos se denominan *AL*, *BL*, *CL* y *DL*, mientras que los bytes de mayor peso se designan mediante *AH*, *BH*, *CH* y *DH*. Estos registros pueden emplearse en la mayoría de las operaciones aritméticas y lógicas, en cualquier manera que el programador especifique, para almacenar los datos antes o después del procesamiento. Asimismo, algunos de estos registros son utilizados específicamente por ciertas instrucciones de programa.

Los registros de puntero y de índice son el puntero de pila (SP), el puntero base (BP), el índice de destino (DI) y el índice de origen (SI). Estos registros se emplean en diversos tipos de operaciones de direccionamiento de memoria bajo el control de la unidad de ejecución (EU).

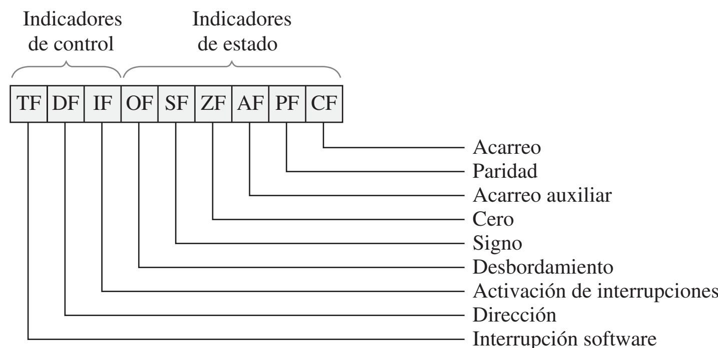


FIGURA 12.12 Los indicadores de estado y de control.

Los indicadores El registro de indicadores contiene nueve bits independientes de estado y de control (**indicadores**), como se muestra en la Figura 12.12. Un indicador de estado es un bit que se utiliza para reflejar una cierta condición después de que la unidad aritmético-lógica haya realizado una operación aritmética o lógica; como ejemplo, podemos citar el bit de acarreo (CF), el bit de resultado cero (ZF) o el bit de signo del resultado (SF). Los indicadores de control se utilizan para modificar las operaciones del procesador en ciertas situaciones.

Modelo software de la familia de procesadores Pentium

A medida que Intel fue introduciendo nuevos microprocesadores, las capacidades y la velocidad de éstos se incrementaron enormemente. Con el procesador Pentium, el concepto pionero de *pipeline* introducido en el 8086/8088 fue ampliado para incorporar dos *pipelines* enteras. El coprocesador externo se incorporó dentro del microprocesador y los buses de datos y de direcciones se ampliaron en gran medida. Otra serie de mejoras (como la velocidad del reloj, la reducción en los ciclos de reloj requeridos por las instrucciones, las capacidades de predicción de salto y la unidad integrada de coma flotante) hacen que el Pentium sea un microprocesador significativamente mejor que sus predecesores. Además de las mejoras relativas al procesador, también se han producido mejoras en otras partes de las computadoras (por ejemplo, en lo que respecta al tamaño y a los protocolos de los buses, a la velocidad, al tamaño de la memoria y al coste). Sin embargo, a pesar de todos estos cambios, los diseñadores de los nuevos procesadores mantuvieron la compatibilidad con el software anterior. En otras palabras, el Pentium más reciente puede seguir ejecutando el software escrito para cualquiera de los procesadores que le precedieron. Esto fue posible porque se mantuvo el modelo software básico (estructura de registros) del microprocesador 8086/8088 original.

Los registros descritos anteriormente para el microprocesador 8086/8088 son un subconjunto de los registros de la familia de procesadores Pentium. A partir del procesador 80386, el conjunto de registros se amplió

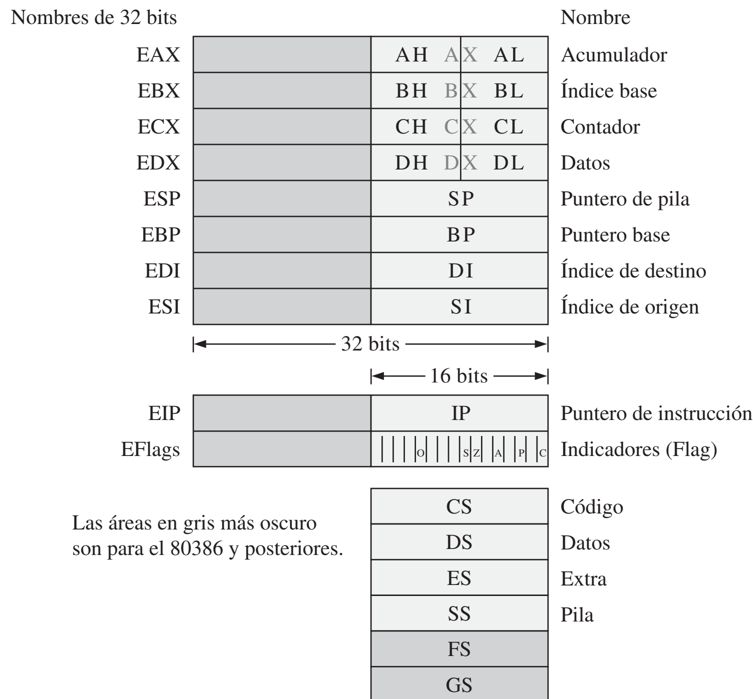


FIGURA 12.13 Registros de los procesadores Intel desde el 8086/8088 hasta el Pentium.

para incluir registros de 32 bits. Los registros de 32 bits mantuvieron los nombres originales, pero añadiendo una E (*Extended*, ampliado) como prefijo de los nombres de registro. Así, la designación de 32 bits para el registro X es EAX. Además, se añadieron dos nuevos registros de segmento. Los registros ampliados se muestran en la Figura 12.13. Las zonas en gris más oscuro representan los registros que sólo están disponibles en el 386 y otros procesadores más recientes.

Además de los registros ampliados, el espacio de memoria direccionable se incrementó enormemente con la introducción de los nuevos procesadores. Para mantener la compatibilidad ascendente, Intel reservó el primer megabyte de memoria para los programas que se ejecutaran en modo real. El **modo real** es cualquier operación que permita al procesador acceder exclusivamente al primer megabyte de memoria, para simular el funcionamiento del 8086/8088. El código escrito para un procesador anterior puede ejecutarse en modo real en un procesador más reciente (aunque el proceso inverso no tiene por qué ser cierto). El código escrito en modo real es generalmente compatible (con algunas excepciones) con todos los procesadores Intel desde el 8086/8088 en adelante.

REVISIÓN DE LA SECCIÓN 12.3

1. Enumerar los registros de propósito general del microprocesador Intel.
2. ¿Cuál es el propósito de la BIU?
3. ¿Se comunica la EU con los buses del sistema?
4. ¿Cuál es la función de la cola de instrucciones?
5. ¿Cuál la ventaja del método segmento:desplazamiento para la formación de direcciones?
6. ¿Qué es el emparejamiento de instrucciones?

12.4 PROGRAMACIÓN DE COMPUTADORAS

El lenguaje ensamblador es una forma de expresar el lenguaje máquina con términos similares a los del idioma inglés, de tal manera que existe una correspondencia biunívoca entre unas instrucciones y otras. El lenguaje ensamblador tiene aplicaciones limitadas y no es portable de un procesador a otro, por lo que la mayoría de los programas informáticos están escritos en lenguajes de alto nivel, tales como C, C++, JAVA, BASIC, COBOL y FORTRAN. Los lenguajes de alto nivel sí son portables y, por tanto, los programas pueden emplearse en distintas computadoras. Los lenguajes de alto nivel deben convertirse a lenguaje máquina para cada microprocesador específico mediante un proceso denominado *compilación.*

Al finalizar esta sección, el lector deberá ser capaz de:

■ Describir algunos conceptos de programación. ■ Explicar los distintos niveles de los lenguajes de programación.

Niveles de los lenguajes de programación

En la Figura 12.14 se muestra un diagrama jerárquico de los lenguajes de programación para computadoras. En el nivel más bajo se encuentra el hardware de la computadora (CPU, memoria, unidad de disco, entrada/salida). A continuación se encuentra el ***lenguaje máquina*** que el hardware comprende, porque está escrito con 1s y 0s (recuerde que una puerta lógica sólo puede reconocer un nivel BAJO, 0, o un nivel ALTO, 1). Por encima del lenguaje máquina se encuentra el ***lenguaje ensamblador*** en el que los 1s y 0s se representan mediante palabras similares a las del idioma inglés. Los lenguajes ensambladores se consideran de bajo nivel porque están estrechamente relacionados con el lenguaje máquina y dependen de la máquina, lo que significa que un determinado lenguaje ensamblador sólo puede emplearse sobre un microprocesador específico.

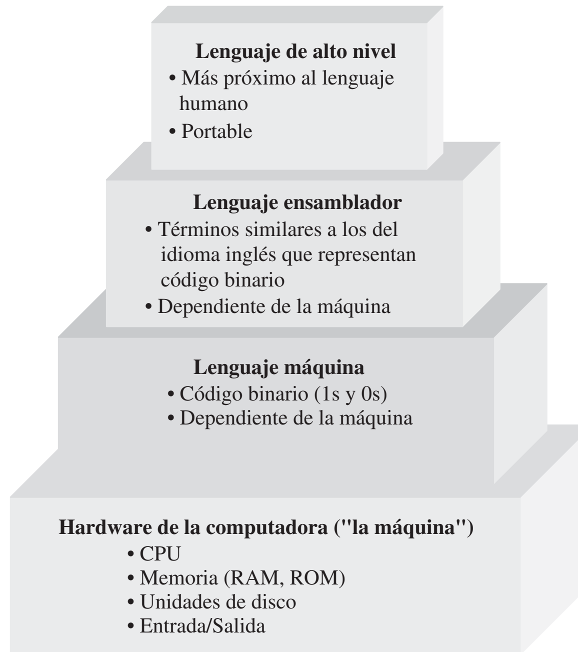


FIGURA 12.14 Jerarquía de los lenguajes de programación, con relación al hardware de una computadora.

Por encima del lenguaje ensamblador se encuentra el ***lenguaje de alto nivel***, que está más próximo al lenguaje humano y se aleja más del lenguaje máquina. Una ventaja del lenguaje de alto nivel sobre el lenguaje ensamblador es que es portable, lo que significa que un programa en lenguaje de alto nivel puede ejecutarse en diversas computadoras. Asimismo, un lenguaje de alto nivel es más fácil de leer, de escribir y de mantener que el lenguaje ensamblador. La mayor parte del software de sistemas (por ejemplo Windows y Unix) y del software de aplicación (por ejemplo, procesadores de texto y hojas de cálculo) está escrito con lenguajes de alto nivel.

Lenguaje ensamblador

Para evitar tener que escribir largas cadenas de 1s y 0s con el fin de representar las instrucciones del microprocesador, se utilizan términos similares a los del idioma inglés, denominados mnemónicos o **códigos de operación**. Cada tipo de microprocesador tiene su propio conjunto de instrucciones mnemónicas que representan códigos binarios. Todas las instrucciones mnemónicas de un microprocesador concreto reciben, colectivamente, el nombre de conjunto de instrucciones. Los lenguajes ensambladores utilizan el conjunto de instrucciones para crear programas para el microprocesador y, puesto que un lenguaje ensamblador está directamente relacionado con el lenguaje máquina (instrucciones en código binario), solemos decir que se trata de un lenguaje de bajo nivel. El lenguaje ensamblador está sólo un paso más allá que el lenguaje máquina.

El lenguaje ensamblador y el correspondiente lenguaje máquina que aquél representa son específicos del tipo de microprocesador concreto o de la familia concreta de microprocesadores. El lenguaje ensamblador no es portable; es decir, no podemos ejecutar un programa en lenguaje ensamblador para un tipo de microprocesador sobre otro tipo distinto de microprocesador. Por ejemplo, un programa de ensamblador para un procesador de Motorola no funcionará sobre los procesadores de Intel. Incluso dentro de una misma familia, los diferentes microprocesadores pueden tener diferentes conjuntos de instrucciones.

Un **ensamblador** es un programa que convierte un programa en lenguaje ensamblador a lenguaje máquina que pueda ser reconocido por el microprocesador. Asimismo, ciertos programas denominados **ensambladores cruzados** traducen un programa en lenguaje ensamblador para un tipo de microprocesador en lenguaje ensamblador para otro tipo de microprocesador.

El lenguaje ensamblador no se suele utilizar para crear grandes programas de aplicación. Sin embargo, el lenguaje ensamblador sí que se emplea a menudo dentro de subrutinas (pequeños programas contenidos dentro de otro programa mayor) que pueden ser invocadas desde un programa escrito en un lenguaje de alto nivel. El lenguaje ensamblador resulta útil en la construcción de subrutinas porque suele ejecutarse más rápido y no tiene ninguna de las restricciones que presentan los lenguajes de alto nivel. El lenguaje ensamblador también se utiliza para el control de máquinas, como por ejemplo en los procesos industriales. Otra área donde se emplea el lenguaje ensamblador es en la programación de juegos.

Conversión de un programa a lenguaje máquina

Todos los programas escritos en cualquier lenguaje ensamblador o en un lenguaje de alto nivel deben convertirse a lenguaje máquina para que una computadora concreta reconozca las instrucciones del programa.

Ensambladores Un ensamblador traduce y convierte un programa escrito en lenguaje ensamblador a código máquina, como se indica en la Figura 12.15. A menudo se utiliza el término **programa fuente** para referirse a un programa escrito en lenguaje ensamblador o de alto nivel. El término **programa objeto** se refiere a la traducción a lenguaje máquina de un programa fuente.

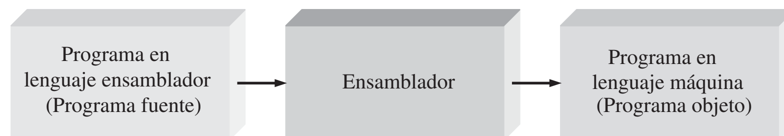


FIGURA 12.15 Conversión de lenguaje ensamblador a lenguaje máquina utilizando un ensamblador.

Compiladores Un compilador es un programa que compila o traduce a código máquina un programa escrito en un lenguaje de alto nivel, como se muestra en la Figura 12.16. El compilador examina el programa fuente completo y recopila y reorganiza las instrucciones. Cada lenguaje de alto nivel incluye un compilador específico para una computadora específica, haciendo que el lenguaje de alto nivel sea independiente de la computadora en la que se utilice. Algunos lenguajes de alto nivel se traducen utilizando lo que se denomina un intérprete, que traduce por separado cada línea de código de programa a lenguaje máquina.



FIGURA 12.16 Conversión de un lenguaje de alto nivel a lenguaje máquina mediante un compilador.

Todos los lenguajes de alto nivel, como C, C++, FORTRAN y COBOL, se ejecutarán en cualquier computadora. Un lenguaje de alto nivel determinado será válido para cualquier computadora, pero el compilador necesario es específico de cada tipo concreto de CPU. Esto se ilustra en la Figura12.17, en la que un mismo programa en lenguaje de alto nivel (escrito en C++ en este caso) se convierte mediante diferentes compiladores específicos de cada máquina.

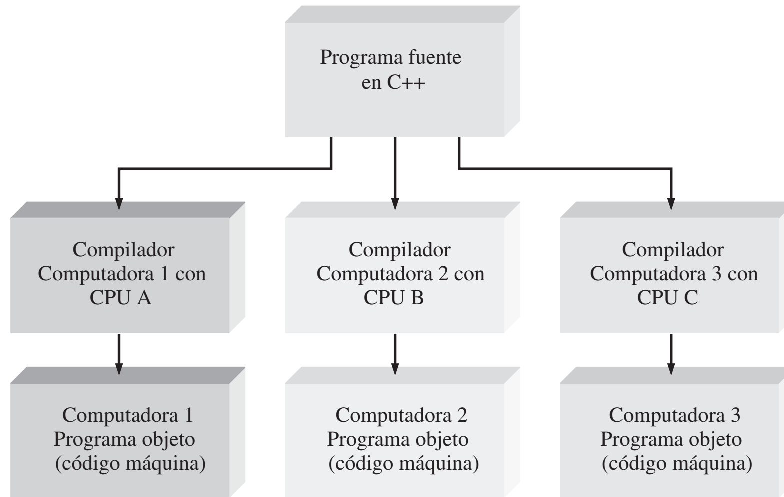


FIGURA 12.17 Independencia con respecto a la máquina de un programa escrito en lenguaje de alto nivel.

Ejemplo de un programa en lenguaje ensamblador Como ejemplo de un programa en lenguaje ensamblador simple, vamos a suponer que queremos que la computadora sume una lista de números almacenados en la memoria y coloque el resultado de nuevo en la memoria. Utilizaremos un cero como último número de la lista para indicar el final de la misma. Los pasos requeridos para llevar a cabo esta tarea son los siguientes:

1. Borrar un registro (en el microprocesador) para almacenar en él el total de la suma de los números.
2. Apuntar al primer número almacenado en la memoria (RAM).
3. Comprobar si el número es cero. Si es cero, quiere decir que se han sumado todos los números.
4. Si el número no es cero, sumar el número de la memoria al total contenido en el registro.
5. Apuntar al siguiente número de la memoria.
6. Repetir los pasos 3, 4 y 5.

A menudo se utiliza un diagrama flujo para mostrar la secuencia de pasos que definen un programa informático. La Figura 12.18 muestra el diagrama de flujo del programa definido por los seis pasos anteriores.

El programa en lenguaje ensamblador implementa el problema de suma mostrado en el diagrama de flujo de la Figura 12.18. Dos de los registros del microprocesador se denominan ax y bx. Los comentarios precedidos por un punto y coma no son reconocidos por el microprocesador y sirven únicamente para explicar el propósito de las instrucciones.

```
      mov ax, 0              ;Sustituye el contenido del registro ax por cero.
                             ;En el registro ax se almacenará el total de la suma.
      mov bx,50H             ;Coloca la dirección hexadecimal de memoria 50
                             ;en el registro bx.
next:cmp word ptr [bx],0     ;Compara con cero el número almacenado en la posición
                             ;de memoria a la que apunta el registro bx
      jz done                ;Si el número de la memoria es cero, saltar a "done"
      add ax, [bx]           ;Sumar el número de la posición de memoria a la que
                             ;apunta bx con el número que se encuentra en el registro
                             ;ax y almacenar la suma en el registro ax.
      add bx,02              ;Sumar 2 a la dirección contenida en el registro bx.
```

```
                             ;Se necesitan dos direcciones para almacenar cada número
                             ;puesto que tienen dos bytes de longitud.
     jmp next                ;Saltar a "next" y repetir el proceso.
done:mov [bx],ax             ;Sustituir el cero almacenado en la última posición
                             ;de memoria a la que apunta el registro bx
                             ;con el total almacenado en el registro ax.
     nop                     ;No operación: indica el final del programa.
```

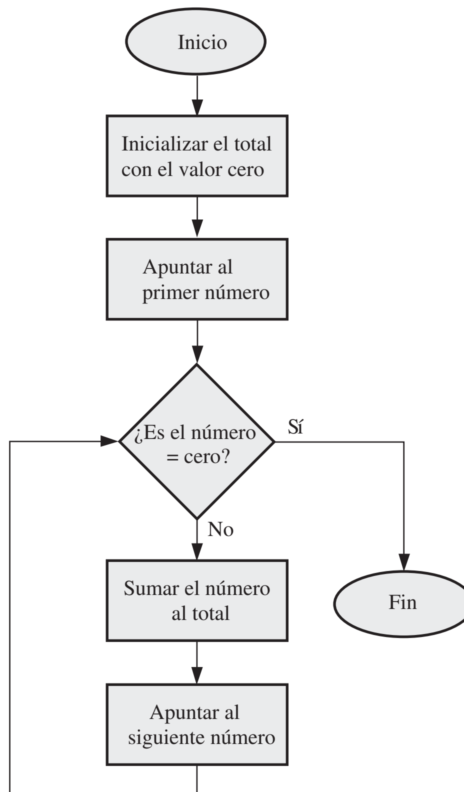


FIGURA 12.18 Diagrama de flujo para sumar una lista de números.

Dependiendo del ensamblador, la mayoría de los programas escritos en lenguaje ensamblador tendrán una serie de directivas de ensamblador que el programa ensamblador utiliza para llevar a cabo diversas tareas. Entre estas tareas se incluyen la configuración de los segmentos, la utilización del conjunto de instrucciones apropiado, la descripción de los tamaños de los datos y la realización de muchas otras tareas de carácter "administrativo". Para simplificar las explicaciones, sólo hemos mostrado en el programa anterior una única directiva (que es obligatoria); esa directiva es **word ptr**, que se utiliza para indicar el tamaño de los datos a los que apunta el registro BX.

El ensamblador Debug

Con unos pequeños cambios podríamos ejecutar el programa anterior, si quisiéramos, utilizando un ensamblador integrado que se incluye en todas las computadoras de tipo PC basadas en DOS. Con ese programa ensamblador podemos observar la ejecución paso a paso. Todas las computadoras de tipo PC basadas en DOS

disponen de un programa llamado *Debug*, que incluye un ensamblador básico. Para utilizar Debug, abra una ventana DOS y escriba **Debug<cr>** en el indicativo DOS (**<cr>** indica un retorno de carro, lo que significa que hay que pulsar la tecla INTRO). Aparecerá un signo menos, que es el indicador del programa Debug. Este programa tiene diversos comandos para consultar o introducir diversos programas. Si escribe **?** en el indicativo de comandos Debug, podrá consultar a la lista completa de comandos del programa. Antes de escribir y ejecutar un programa en ensamblador, podemos introducir algunos datos escribiendo la información que a continuación mostramos en negrita:

```
-a 50
```

Este comando le dice a Debug que comience a ensamblar las instrucciones en el segmento de datos actual empleando un desplazamiento de valor 50H. Aunque lo que vamos a introducir son datos, esta es la manera más simple de introducir una palabra de 16 bits (recuerde que todos los datos dentro de Debug se especifican en hexadecimal). Debug responderá con una dirección de segmento que será, sin ninguna duda, diferente de la que aquí mostramos (20D8), pero esto no importa. La dirección de desplazamiento (50H) será la misma. Escriba la información mostrada en negrita (recuerde que cada <cr> quiere decir que hay que pulsar la tecla INTRO).

```
20D8:0050 dw 30 <cr>
20D8:0052 dw 15 <cr>
20D8:0054 dw a0 <cr>
20D8:0056 dw 0c <cr>
20D8:0058 dw 00 <cr>
<cr>
```

La palabra **dw** es una directiva del ensamblador. Esa directiva no se almacena, sino que simplemente indica al ensamblador que cada dato tiene dos bytes de longitud. En los datos mostrados, sólo se emplea un byte, pero el programa guardará cada dato utilizando dos posiciones de memoria incluyendo un cero en la posición de mayor peso.

Ahora podemos introducir un programa en la posición 100 de la siguiente forma:

```
-a 100 <cr>
20D8:0100 mov ax,0 <cr>
20D8:0103 mov bx,50 <cr>
20D8:0106 cmp word ptr [bx],0 <cr>
20D8:0109 jz 112 <cr>
20D8:010B add ax,[bx] <cr>
20D8:010D add bx,2 <cr>
20D8:0110 jmp 106 <cr>
20D8:0112 mov [bx],ax <cr>
20D8:0114 nop <cr>
20D8:0115 <cr>
```

Para confirmar que el programa ha sido introducido correctamente, podemos escribir **u 100 114** en el indicativo de comandos de Debug y el código introducido se mostrará en la pantalla (aparecerá en letras mayúsculas). Ahora escriba **r** en el indicativo de Debug y se mostrará una lista de los registros de 16 bits y del estado de los indicadores. Observe que el registro IP debe tener el valor 100, que es la dirección de inicio del código. Por debajo de la lista de registros, se mostrará la primera instrucción (MOV AX, 0000). Podemos hacer que Debug ejecute esta instrucción mediante el comando **t** (*trace*, traza). Esto hará que se muestre en pantalla el estado de todos los registros junto con la siguiente instrucción (MOV BX, 0050). Si ejecutamos ésta mediante un comando **t**, veremos que el número 0050 ha sido almacenado en el registro BX. Los pasos hasta este punto se muestran en la Figura 12.19. Observe que el primer dato se muestra en la esquina inferior derecha.

```
–u 100 114
20D8:0100 B80000         MOV     AX,0000
20D8:0103 BB5000         MOV     BX,0050
20D8:0106 833F00         CMP     WORD PTR [BX],+00
20D8:0109 7407           JZ      0112
20D8:010B 0307           ADD     AX,[BX]
20D8:010D 83C302         ADD     BX,+02
20D8:0110 EBF4           JMP     0106
20D8:0112 8907           MOV     [BX],AX
20D8:0114 90             NOP
–r
AX=0000  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=20D8  ES=20D8  SS=20D8  CS=20D8  IP=0100   NV UP EI PL ZR NA PE NC
20D8:0100 B8000          MOV     AX,0000
–t

AX=0000  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=20D8  ES=20D8  SS=20D8  CS=20D8  IP=0103   NV UP EI PL ZR NA PE NC
20D8:0103 BB5000         MOV     BX,0050
–t

AX=0000  BX=0050  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=20D8  ES=20D8  SS=20D8  CS=20D8  IP=0106   NV UP EI PL ZR NA PE NC
20D8:0106 833F00         CMP     WORD PTR [BX],+00                DS:0050=0030
–
```

FIGURA 12.19 Pasos iniciales de ejecución del programa de suma mediante Debug.

```
DS=20D8  ES=20D8  SS=20D8  CS=20D8  IP=0110   NV UP EI PL NZ NA PO NC
20D8:0110 EBF4           JMP     0106
–t

AX=00F1  BX=0058  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=20D8  ES=20D8  SS=20D8  CS=20D8  IP=0106   NV UP EI PL NZ NA PO NC
20D8:0106 833F00         CMP     WORD PTR [BX],+00                DS:0058=0000
–t

AX=00F1  BX=0058  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=20D8  ES=20D8  SS=20D8  CS=20D8  IP=0109   NV UP EI PL ZR NA PE NC
20D8:0109 7407           JZ      0112
–t

AX=00F1  BX=0058  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=20D8  ES=20D8  SS=20D8  CS=20D8  IP=0112   NV UP EI PL ZR NA PE NC
20D8:0112 8907           MOV     [BX],AX                          DS:0058=0000
–t

AX=00F1  BX=0058  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=20D8  ES=20D8  SS=20D8  CS=20D8  IP=0114   NV UP EI PL ZR NA PE NC
20D8:0114 90             NOP
–d 0050 005f
20D8:0050   30 00 15 00 A0 00 0C 00-F1 00 00 00 00 20 20 20   0...........
–
            Datos originales          Suma
```

FIGURA 12.20 Última parte del proceso de traza de la ejecución del programa de suma. La suma 00F1 se muestra en la última línea, apareciendo primero la parte menos significativa (F1) de la suma.

Continuando de esta forma, podemos ejecutar el código completo y observar los cambios en los registros a medida que el microprocesador pasa por las distintas instrucciones, como se muestra en la Figura 12.20. Este programa tiene una estructura de programación bastante común, que se denomina *bucle*. Un bucle es un grupo repetitivo de instrucciones que se ejecutan hasta que se cumple una determinada condición, en este caso, la condición es encontrar un cero dentro de los datos. Después de haber encontrado un cero, se ejecutará la última instrucción y se almacenará la suma (en este caso, el valor hexadecimal de la suma es 00F1) en el lugar del cero que indicaba el final de los datos. Podemos observar esto escribiendo **d 0050 005F** (para visualizar el contenido de la memoria entre las direcciones 0050 y 005F) en el indicativo de comandos de Debug, después de alcanzar la última instrucción (NOP), como se muestra en la Figura 12.20. El resultado aparece en los bytes 9 y 10 (quinta palabra), en la línea que se muestra a continuación de la instrucción mediante la que hemos indicado que deseábamos visualizar la memoria. Observe que se muestra primero la parte menos significativa de la respuesta. Cuando el microprocesador ejecuta este programa en "tiempo real" requiere menos de 1 μs para completar el proceso completo. Si quisiéramos repetir el proceso tendríamos que volver a almacenar un cero en la posición 0058, porque el programa lo ha sustituido por la suma (recuerde que el programa utiliza el cero como forma de detectar que ha llegado al final de los datos).

EJEMPLO 12.2

Escribir las instrucciones de un programa en lenguaje ensamblador que halle el número sin signo de mayor valor dentro de los datos y lo almacene en la última posición. Suponga que el final de la lista de datos se indica mediante un cero.

Solución El diagrama de flujo se muestra en la Figura 12.21.
Suponemos que los datos son los mismos que antes. El listado del programa (con comentarios) es el siguiente:

```
       mov ax,0000         ; el valor inicial de BIG está en el registro ax
       mov bx,0050         ; apuntar a la posición de memoria (50H) donde
                           ; empiezan los datos
repeat:cmp [bx],ax         ; ¿es el dato mayor que BIG?
       jbe check           ; si el dato es más pequeño, ir a "check"
       mov ax,[bx]         ; en caso contrario, almacenar el nuevo máximo en ax
check: add bx,02           ; apuntar al siguiente número en memoria
                           ;(dos bytes por palabra)
       cmp word ptr [bx],0 ; comprobar si es el último dato
       jnz repeat          ; continuar si el dato es distinto de cero
       mov [bx],ax         ; guardar BIG en memoria
       nop                 ; no operación
```

En la Figura 12.22 se muestra el listado de este programa obtenido mediante Debug. Los datos se introducen de la misma forma que antes comenzando en la posición 0050. En este caso se utilizan los mismos datos, pero podríamos introducir nuevos datos si quisiéramos. Es importante introducir un cero como último dato porque el programa continuará hasta que encuentre ese valor. El cero es sustituido por el valor máximo de los datos cada vez que se ejecuta el programa. El programa se introduce comenzando el ensamblado en la posición 100, introduciendo el programa después de ejecutar el comando **a 100**.
Podemos hacer una traza del programa para ver su ejecución, analizando ésta paso a paso. Alternativamente, podemos escribir el comando **g = 100 116** para

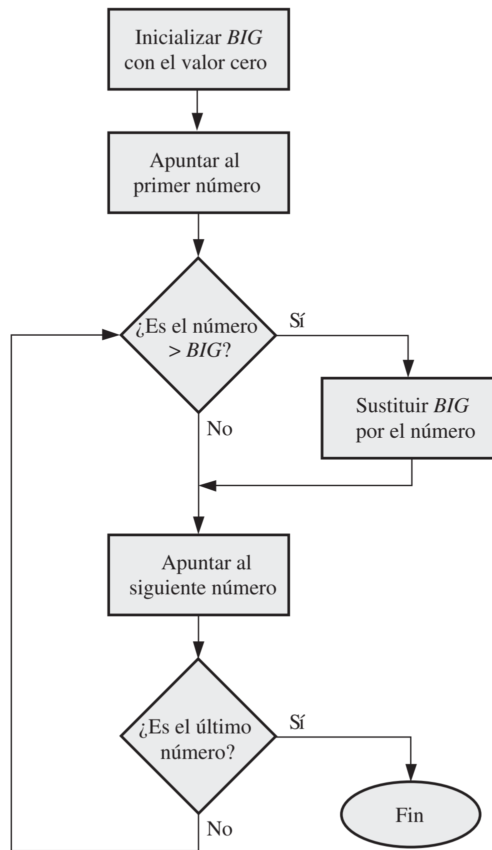


FIGURA 12.21 Diagrama de flujo. La variable BIG representa el valor máximo.

```
–a 100
20D8:0100 mov ax,0
20D8:0103 mov bx,50
20D8:0106 cmp [bx],ax
20D8:0108 jbe 10c
20D8:010A mov ax, [bx]
20D8:010C add bx,2
20D8:010F cmp word ptr [bx],0
20D8:0112 jnz 106
20D8:0114 mov [bx],ax
20D8:0116 nop
20D8:0117
```

FIGURA 12.22 Listado del programa obtenido mediante Debug.

ejecutar las instrucciones comprendidas entre la dirección 100 y la 116. La Figura 12.23 muestra los datos antes y después de la ejecución. Observe que cada dato se almacena utilizando dos bytes (aunque el propio dato sólo tiene un byte), porque hemos definido los datos como palabras. Asimismo, observe

que el byte de menor peso se almacena en la memoria antes que el byte de mayor peso. Después de ejecutar el programa, vemos que el último de los datos (el que tenía el valor cero) ha sido sustituido por el valor máximo (en este caso, A0). Podemos ver que este valor está almacenado tanto en el registro AX como en la memoria.

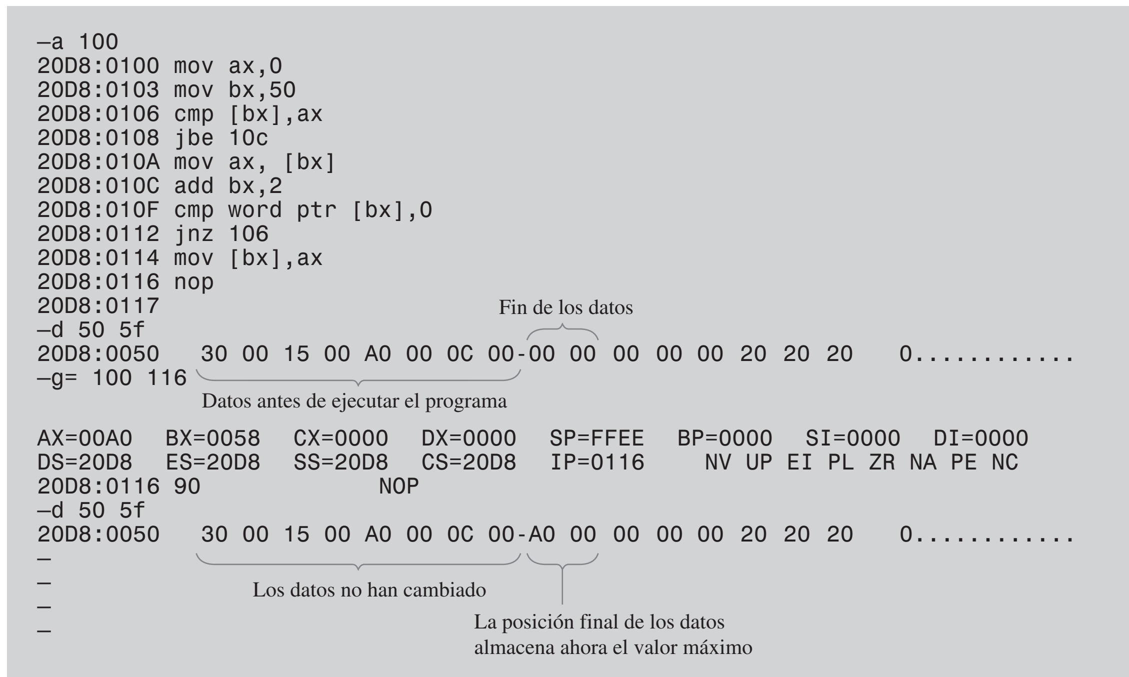


FIGURA 12.23 Datos antes y después de la ejecución.

Problema relacionado Explicar cómo habría que modificar el diagrama de flujo para determinar el número mínimo de la lista en lugar del máximo.

Tipos de instrucciones

Los programas de esta sección sólo muestran unos cuantos de los centenares de variantes de instrucciones disponibles para los programadores. Para simplificar el aprendizaje del conjunto de instrucciones de Intel, esas instrucciones se dividen en siete categorías, las cuales se describen a continuación.

Transferencia de datos Ya hemos presentado la más básica de las instrucciones de transferencia de datos MOV, en los programas de ejemplo. La instrucción MOV puede utilizarse, por ejemplo, de distintas formas para copiar un byte, una palabra (16 bits) o una doble palabra (32 bits) entre diversos orígenes y destinos, como por ejemplo registros, memoria y puertos de E/S. Un mnemónico mejor para MVO podría haber sido "COPY" (copiar) porque esto es lo que en realidad hace la instrucción, copiar los datos no moverlos. Otras instrucciones de transferencia de datos son, por ejemplo, IN (obtener datos de un puerto), OUT (enviar datos a un puerto), PUSH (copiar datos a la pila, un área separada de la memoria), POP (copiar datos de la pila) y XCHG (intercambio).

Aritméticas Hay varias instrucciones y variantes para la suma, resta, multiplicación y división de valores. Ya hemos utilizado la instrucción ADD para las sumas en ambos programas de ejemplo. Otras instrucciones aritméticas son, por ejemplo, INC (incrementar), DEC (decrementar), CMP (comparar), UB (sustraer, restar), MUL (multiplicar) y DIV (dividir). Las distintas variantes de estas instrucciones permiten realizar operaciones de acarreo, así como llevar a cabo operaciones aritméticas con signo como sin signo. Estas instrucciones permiten especificar operandos localizados en la memoria, en los registros y en los puertos de E/S.

Manipulación de bits Este grupo de instrucciones incluye las utilizadas para tres clases de operaciones: operaciones lógicas (booleanas), desplazamientos y rotaciones. Las instrucciones lógicas son: NOT, AND, OT, XOR y TEST. Un ejemplo de una instrucción de desplazamiento es SAR (*Shift Arithmetic Right*, desplazamiento aritmético a la derecha). Un ejemplo de una instrucción de rotación es ROL (*Rotate Left*, rotación a la izquerda). Cuando los bits se extraen por desplazamiento de un operando se pierden, pero si se extraen por rotación vuelven a introducirse en el operando por el otro extremo. Estas instrucciones lógicas, de desplazamiento y de rotación pueden operar tanto sobre bytes como sobre palabras almacenados tanto en registros como en la memoria.

Bucles y saltos Estas instrucciones están diseñadas para alterar la secuencia normal (una detrás de otra) de ejecución de las instrucciones. La mayoría de estas instrucciones consulta los indicadores del procesador para determinar qué instrucción hay que procesar a continuación. En el Ejemplo 12.2 hemos utilizado las instrucciones JBE y JNZ para alterar la secuencia de ejecución. Entre las otras instrucciones de este grupo podemos citar JMP (salto incondicional), JA (salto hacia arriba), JO (saltar desbordamiento), LOOP (decrementar el registro CX y repetir si es distinto de cero) y muchas otras.

Procesamiento de cadenas Una **cadena** es una secuencia **contigua** (uno después de otro) de bytes o palabras. Las cadenas son bastante comunes en los programas informáticos. Un ejemplo sencillo sería una frase que el programador quiera mostrar en pantalla. Hay cinco instrucciones básicas para cadenas que están diseñadas para copiar, cargar, almacenar, comparar o analizar una cadena, bien de byte en byte, o de palabra en palabra. Como ejemplos de instrucciones de procesamiento de cadenas tenemos MOVSB (copiar una cadena de byte en byte) y MOVSW (copiar una cadena de palabra en palabra).

Subrutinas e interrupciones Una **subrutina** es un miniprograma que puede utilizarse repetidamente aunque sólo es necesario programarlo una vez. Por ejemplo, si un programador necesita convertir números ASCII de un teclado a formato BCD, una estructura de programación simple consistiría en definir las instrucciones necesarias en un proceso separado e "invocar" dicho proceso cada vez que sea necesario. Entre las instrucciones de este grupo podemos citar CALL (iniciar una subrutina) y RET (volver al programa principal).

Control del procesador Éste es un pequeño grupo de instrucciones que permite controlar de manera directa algunos de los indicadores del procesador, así como llevar a cabo otra serie de tareas misceláneas. Un ejemplo es la instrucción STC (activar el indicador de acarreo).

Programación de alto nivel

Los pasos básicos que hay que dar cuando se escribe un programa de alto nivel, independientemente del lenguaje de programación concreto que se utilice, son los siguientes:

1. Determinar y especificar el problema que hay resolver o la tarea que hay que realizar.
2. Crear un algoritmo, es decir, desarrollar una serie de pasos para llevar a cabo la tarea.
3. Expresar esos pasos utilizando un lenguaje de programación concreto e introducirlos en el editor de texto utilizado para escribir el software.
4. Compilar (o ensamblar) y ejecutar el programa.

Vamos a mostrar un ejemplo de programación de alto nivel mediante un programa simple. El siguiente programa C++ implementa el mismo problema de la suma definido en el diagrama de flujo de la Figura 12.18 y que ya hemos implementado anteriormente utilizando lenguaje ensamblador.

```
int total = 0;             //Inicializar el total a 0.
while (*number ! = 0X00)   //Repetir el bucle mientras no se encuentre el valor.
                           //El asterisco situado antes del puntero number indica
                           //que se está evaluando el contenido de la posición
                           //memoria a la que apunta el identificador number.
{
total = total + *number;   //Suma acumulativa del total.
number++;                  //Incrementar el puntero para acceder al siguiente número
                           //de la memoria.
}
```

Este programa C++ es equivalente al programa ensamblador que suma una serie de números y da como resultado un valor total.

```
in total = 0;                               mov ax, 0
while (*number ! = 0X00)                    mov bx, 50H
{                                      next:cmp word ptr [bx], 0
   total = total + *number;                 jz done
number++;                                   add ax, [bx]
}                                           add bx, 02
                                            jmp next
                                       done:mov [bx], ax
                                            nop
```

C++ → Equivalente → Ensamblador

REVISIÓN DE LA SECCIÓN 12.4

1. Definir *programa.*
2. ¿Qué es un código de operación?
3. ¿Qué es una cadena?

12.5 INTERRUPCIONES

En esta sección, presentamos el tema del establecimiento de comunicaciones entre un periférico y la CPU. Se exponen tres métodos distintos: E/S por sondeo, E/S controlada por interrupciones e interrupciones software.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Explicar la necesidad de utilizar interrupciones en un sistema informático. ■ Describir el concepto básico de E/S por sondeo. ■ Describir el concepto básico de E/S controlado por interrupciones. ■ Explicar el concepto de interrupción software.

En los sistemas basados en microprocesador, como las computadoras personales, los dispositivos periféricos requieren que la CPU les preste servicio periódicamente. El término *servicio* significa generalmente enviar datos hacia el dispositivo, extraer datos del dispositivo o realizar algún tipo de proceso de actualiza-

ción. Hay tres formas mediante las que se puede iniciar una rutina de servicio: *E/S por sondeo, E/S controlada por interrupciones* o *interrupciones software.* Recuerde que una interrupción es una señal o instrucción que hace que el proceso actual se detenga temporalmente mientras se ejecuta una rutina de servicio.

En general, los dispositivos periféricos son muy lentos comparados con la CPU. Una impresora puede que sólo sea capaz de imprimir por término medio unos pocos caracteres por segundo (un carácter está representado por 8 bits), dependiendo del tipo de material que se esté imprimiendo y del tipo de impresora. La tasa de entrada a través del teclado puede ser de sólo uno o dos caracteres por segundo, dependiendo de la velocidad del operador. Por tanto, entre los instantes en que la CPU debe dar servicio a un periférico, puede realizar una gran cantidad de procesamiento. En la mayoría de los sistemas, este tipo de procesamiento debe maximizarse utilizando un método eficiente para prestar servicio a los periféricos.

E/S por sondeo

Uno de los métodos para dar servicio a los periféricos se denomina **sondeo**. Según este método, la CPU debe consultar secuencialmente a cada dispositivo periférico a ciertos intervalos, para ver si necesita servicio o está listo para el mismo. La Figura 12.24 ilustra el método básico de E/S por sondeo.

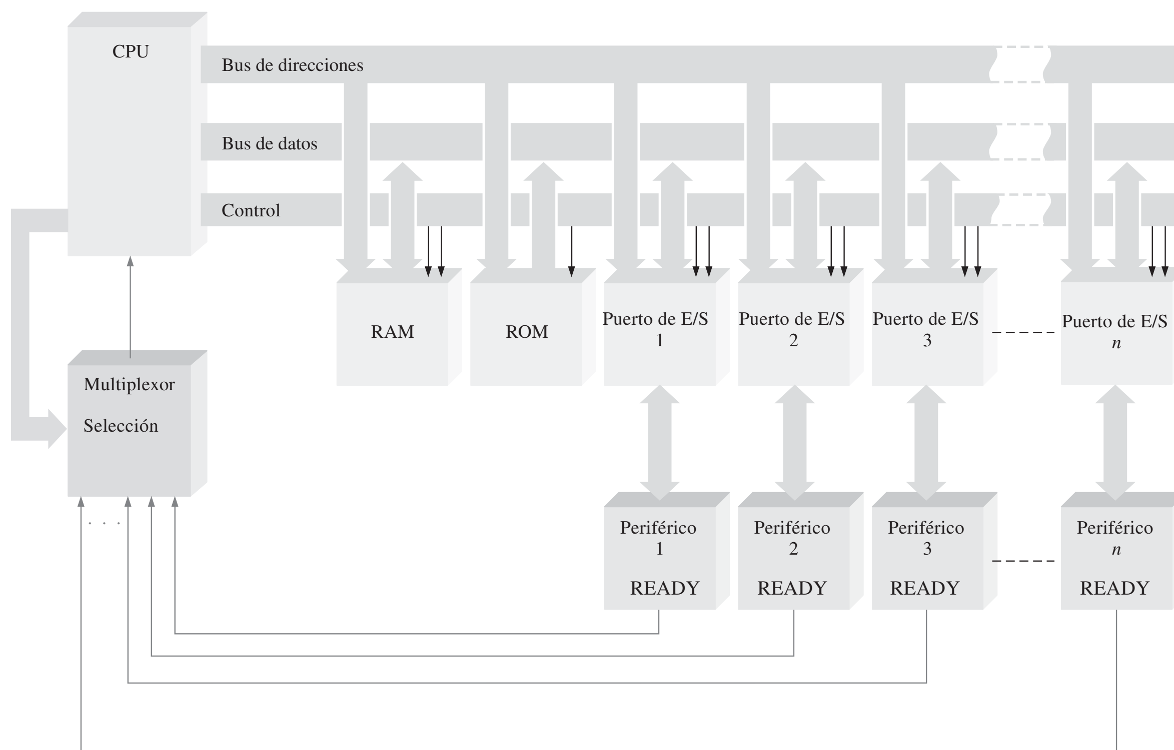


FIGURA 12.24 Configuración básica de E/S por sondeo.

La CPU selecciona secuencialmente cada dispositivo periférico a través del multiplexor para ver si necesita servicio, consultando su línea READY (preparado). Ciertos periféricos pueden necesitar que se les preste servicio a intervalos irregulares e impredecibles, es decir, pueden necesitar que se les preste servicio más frecuentemente en unas ocasiones que en otras. Sin embargo, la CPU deberá sondear al dispositivo a la frecuencia más alta posible. Por ejemplo, supongamos que un cierto periférico necesita que se le preste servicio ocasionalmente cada 1000 μs, aunque la mayor parte de las veces sólo requiere que se le preste servicio cada

100 ms. Como puede comprenderse fácilmente se desperdicia un valioso tiempo de procesamiento si la CPU tiene que sondear al dispositivo, como es su obligación, a la frecuencia máxima (cada 1000 μs), porque la mayor parte de las veces el dispositivo informará de que no necesita ser atentido en el momento de realizarse el sondeo.

Cada vez que la CPU sondea un dispositivo, debe detener el programa que esté procesando actualmente, ejecutar la secuencia de sondeo, proporcionar servicio en caso necesario y luego volver al punto en el que interrumpió el programa actual.

Otro problema con la técnica de E/S por sondeo secuencial es que si dos o más dispositivos necesitan ser atendidos al mismo tiempo, el dispositivo que será atendido en primer lugar es aquel que haya sido sondeado primero, los otros dispositivos tendrán que esperar, aun cuando puedan necesitar ser atendidos con mucha mayor urgencia que el primer dispositivo sondeado. Como puede ver, el sondeo sólo es adecuado para dispositivos a los que se pueda dar servicio a intervalos regulares y predecibles, y sólo en aquellas situaciones en las que no tengan que tenerse en cuenta las consideraciones de prioridad.

E/S controlada por interrupciones

Esta técnica resuelve las desventajas del método de sondeo. Con el método controlado por interrupciones, la CPU responde a una solicitud de servicio sólo cuando un dispositivo periférico efectúa su solicitud de manera explícita. De este modo, la CPU puede concentrarse en ejecutar el programa actual, sin tener que detenerlo innecesariamente para ver si un dispositivo necesita ser atendido.

Cuando la CPU recibe una señal de interrupción de E/S, detiene temporalmente el programa actual, confirma la interrupción y extrae de la memoria un programa especial (rutina de atención de la interrupción)

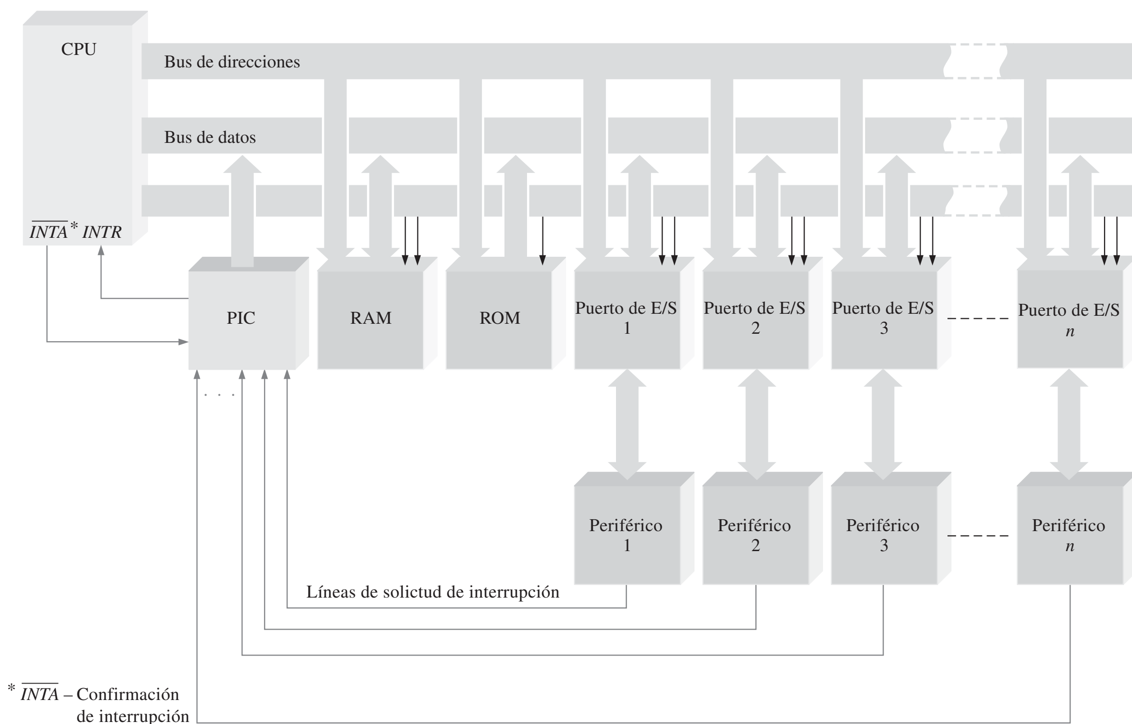


FIGURA 12.25 Configuración básica de E/S controlada por interrupciones.

adaptado al dispositivo concreto que haya generado la interrupción. Una vez generada la rutina de atención a la interrupción, la CPU continúa con aquello que estuviera haciendo. Un dispositivo especial denominado controlador de interrupciones programable (**PIC,** *Programmable Interrupt Controller*) gestiona las interrupciones de acuerdo con un mecanismo de prioridad. Este dispositivo acepta las solicitudes de servicio procedentes de los periféricos. Si dos o más dispositivos solicitan servicio al mismo tiempo, aquél que tenga asignada la prioridad más alta será servido primero, después el que tenga la siguiente prioridad más alta y así sucesivamente. Después de enviar una señal de interrupción (*INTR*) a la CPU, el controlador PIC proporciona a la CPU la información necesaria para "dirigir" a la CPU hacia la dirección de memoria inicial de la rutina de atención a la interrupción apropiada. Este proceso se denomina *vectorización*. La Figura 12.25 muestra una configuración básica de E/S controlada por interrupciones.

Interrupciones software

Otro tipo de interrupción es la denominada **interrupción software**. Las interrupciones software son interrupciones de programa que pueden invocar las mismas rutinas de servicio que hemos descrito anteriormente. La diferencia es que esas rutinas se invocan desde el software en lugar de invocarse desde un hardware externo. Al ser invocada la rutina de atención a la interrupción se ejecuta exactamente como si hubiera tenido lugar una interrupción hardware. Las cinco primeras interrupciones software están definidas por Intel, otras están definidas por el sistema BIOS y por DOS para llevar a cabo muchas de las operaciones de E/S, como leer y escribir datos en disco, escribir datos en la pantalla o leer datos procedentes del teclado.

REVISIÓN DE LA SECCIÓN 12.5

1. ¿En qué se diferencia la E/S controlada por interrupciones de la E/S por sondeo?
2. ¿Cuál es la ventaja principal de la E/S controlada por interrupciones?
3. ¿Qué es una interrupción software?

12.6 ACCESO DIRECTO A MEMORIA (DMA)

En esta breve sección, se define la técnica de transferencia de datos denominada acceso directo a memoria (DMA, *Direct memory access*), presentándose una comparación entre las transferencias realizadas por la CPU y las transferencias por DMA.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Definir el término *DMA*. ■ Comparar una transferencia de datos de E/S de memoria realizada por la CPU y una transferencia DMA.

Todas las transferencias de datos de E/S estudiadas hasta el momento se han hecho a través de la CPU. Por ejemplo, cuando los datos se transfieren desde la RAM a un dispositivo periférico, la CPU lee el primer byte de datos de la memoria y lo carga en un registro interno del microprocesador. Después, la CPU escribe el byte de datos en el correspondiente puerto de E/S. Esta operación de lectura/escritura se repite para cada byte del grupo de datos que haya que transferir. La Figura 12.26 ilustra este proceso.

Para bloques de datos largos, los pasos intermedios del microprocesador consumen mucho tiempo. Por esta razón, muchos sistemas utilizan la técnica denominada **DMA** (*Direct Memory Access*, acceso directo a memoria) para acelerar las transferencias entre la RAM y determinados dispositivos periféricos. Básicamente, para ciertos tipos de transferencias de datos, el acceso directo a memoria ignora a la CPU, eliminando el tiempo consumido por los ciclos de carga y ejecución normales requeridos para cada operación de lectura o escritura.

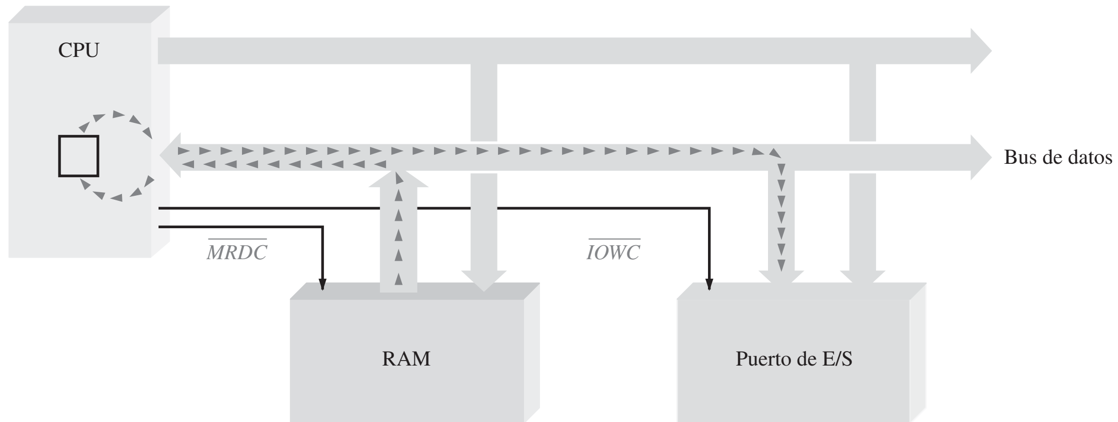


FIGURA 12.26 Transferencia de E/S de memoria realizada por la CPU.

Para la transferencia directa de memoria se utiliza un dispositivo denominado controlador DMA, que toma el control de los buses del sistema y permite que los datos fluyan directamente entre la RAM y los dispositivos periféricos, como muestra la Figura 12.27. En particular, las transferencias entre la unidad de disco y la RAM se hacen con este método, debido a la gran cantidad de datos implicados y a la naturaleza secuencial de las transferencias. El controlador de DMA puede efectuar trasferencias de datos varias veces más rápidamente que la CPU.

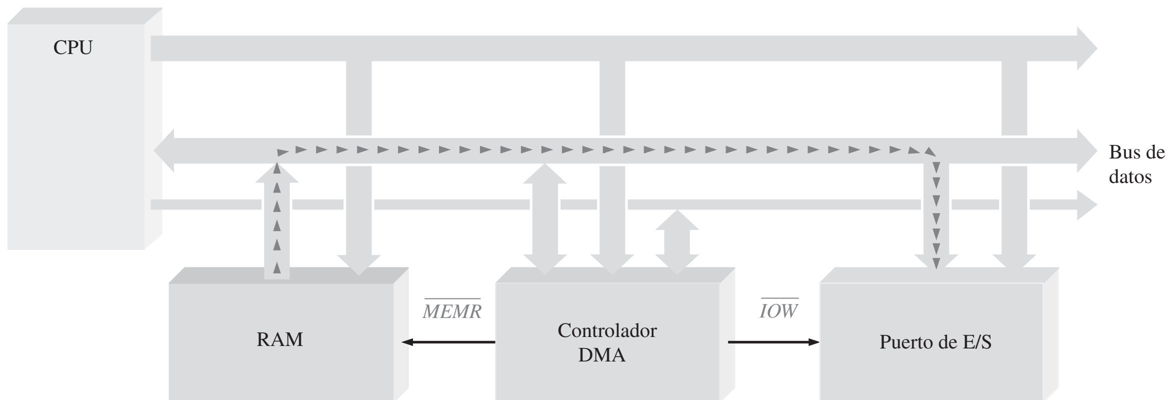


FIGURA 12.27 Transferencia por acceso directo a memoria.

REVISIÓN DE LA SECCIÓN 12.6

1. ¿Qué significa DMA?
2. Explicar la ventaja de las transferencias mecanismo DMA y dar un ejemplo de un tipo de transferencia para el que se utilice frecuentemente esta técnica.

12.7 INTERFACES INTERNAS

Como hemos visto, todos los componentes de una computadora se interconectan mediante buses, que actúas como rutas de comunicación. Físicamente, un bus es un conjunto de hilos conductores que sirve

para interconectar dos o más componentes funcionales de un sistema o de varios sistemas distintos. Eléctricamente, un bus es una colección de niveles de tensión y/o corriente especificados y de señales que permiten que los diversos dispositivos conectados al bus puedan funcionar conjutamente.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Explicar el concepto de bus multiplexado. ■ Explicar la razón de la existencia de salidas triestado.

Buses multiplexados básicos

En las computadora, el microprocesador controla (y se comunica con) las memorias y los dispositivos de entrada/salida (E/S) a través de la *estructura de bus interna*, como se indica en la Figura 12.28. El bus está multiplexado de manera que cualquiera de los dispositivos que están conectados al mismo pueda enviar o recibir datos hacia o desde los otros dispositivos. El dispositivo transmisor se denomina a menudo **origen**, mientras que el dispositivo receptor se llama comúnmente **aceptor**. En cualquier momento dado, sólo puede haber un origen activo. Por ejemplo, la RAM puede estar enviando datos a la interfaz de entrada/salida (E/S) bajo control del microprocesador.

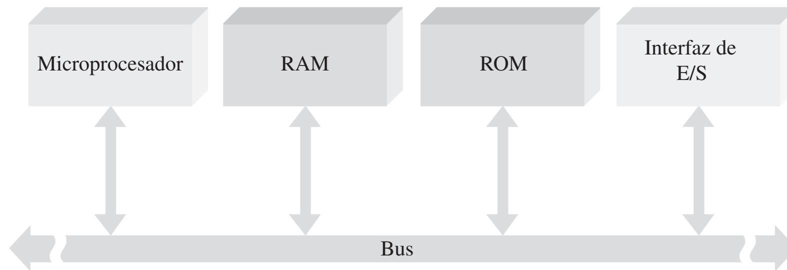


FIGURA 12.28 Interconexión de componentes de un sistema basado en microprocesador mediante un bus bidireccional multiplexado.

Señales del bus

Utilizando la técnica de control síncrono del bus, el microprocesador suele ser el encargado de generar todas las señales de temporización y control. Entonces, los otros dispositivos sincronizan sus operaciones con dichas señales de control y temporización. Con la técnica de control asíncrono del bus, las señales de control y temporización son generadas conjuntamente por un origen y un aceptor. El proceso de establecer conjuntamente la comunicación se denomina **negociación**. En la Figura 12.29 se muestra un ejemplo simple de secuencia de negociación.

Una función de control importante se denomina **arbitraje del bus**. El arbitraje evita que dos orígenes traten de utilizar el bus al mismo tiempo.

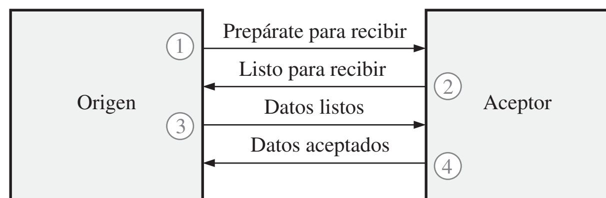


FIGURA 12.29 Un ejemplo de secuencia de negociación.

Conexión de dispositivos a un bus

Normalmente se emplean *buffers* ***triestado*** para conectar las salidas de un dispositivo de origen a un bus. Usualmente, existe más de un origen conectado a un mismo bus pero sólo uno de ellos puede tener acceso al bus en cada momento. Los demás orígenes deben desconectarse del bus para evitar la denominada **contienda de bus**.

Los circuitos triestado se utilizan para conectar un origen a un bus o para desconectarlo del bus, como se ilustra en la Figura 12.30(a) para el caso de dos orígenes. La entrada de selección se emplea para conectar el

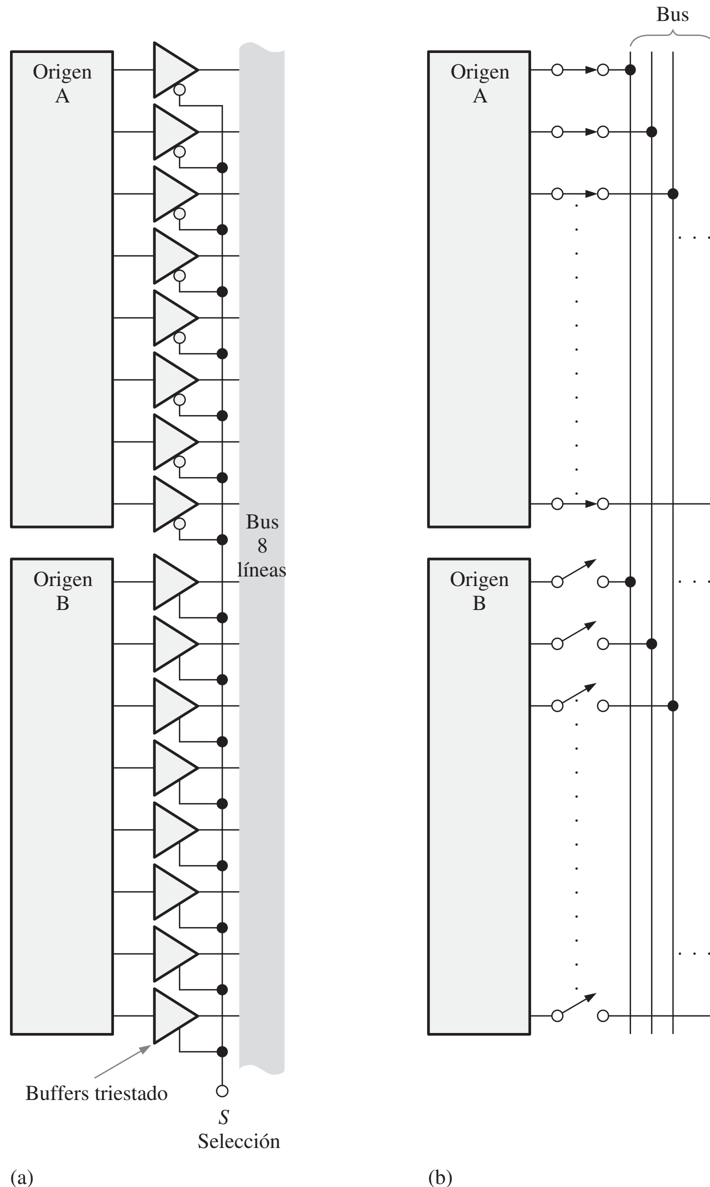


FIGURA 12.30 Interfaz con un bus mediante *buffers* triestado.

origen A o el origen B, pero no ambos al mismo tiempo, al bus compartido por los dos. Cuando la entrada de selección está a nivel BAJO, el origen A está conectado y el origen B está desconectado. Cuando la entrada de selección está a nivel ALTO, el origen A está desconectado y el origen B está conectado. En la parte (b) de la figura se muestra un equivalente mediante conmutadores de esta acción.

Cuando la entrada de habilitación de un circuito triestado no está activa, el dispositivo se encuentra en un estado de alta impedancia (**alta-Z**) y actúa como un conmutador abierto. Muchos circuitos integrados digitales proporcionan *buffers* triestado internos en sus líneas de salida. Una salida triestado se indica mediante un símbolo ∇, como se muestra en la Figura 12.31.

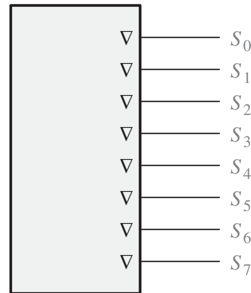


FIGURA 12.31 Método para indicar las salidas triestado en un circuito integrado.

Operación de un buffer triestado La Figura 12.32(a) muestra el símbolo lógico para un *buffer* triestado no inversor con una entrada de habilitación activa a nivel ALTO. La parte (b) de la figura muestra otro *buffer* similar con una entrada de habilitación activa a nivel BAJO.

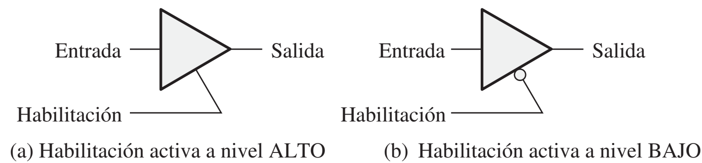


FIGURA 12.32 Símbolos de *buffers* triestado.

La operación básica de un *buffer* triestado puede comprenderse en términos de una acción de conmutación, como se ilustra en la Figura 12.33. Cuando la entrada de habilitación está activa, la puerta opera como un circuito no inversor normal. Es decir, la salida está a nivel ALTO cuando la entrada está a nivel ALTO y a nivel BAJO cuando la entrada está a nivel BAJO, como se muestra en las partes (a) y (b), respectivamente. Los niveles ALTO y BAJO representan dos de los estados. El *buffer* operará en su tercer estado cuando la entrada de habilitación no esté activa. En este estado, el circuito actúa como un conmutador abierto y la salida está completamente desconectada de la entrada, como se muestra en la parte (c). Esto se denomina en ocasiones estado de *alta impedancia* o de *alta Z.*

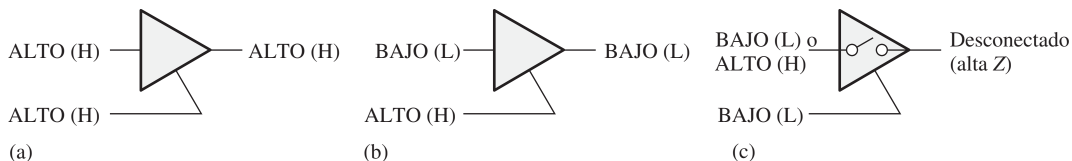


FIGURA 12.33 Operación del buffer triestado.

Muchos microprocesadores, memorias y otras funciones de circuito integrado disponen de *buffer* triestado que sirve para implementar la interfaz con los buses. Tales *buffers* son necesarios cuando hay dos o más dispositivos conectados a un bus común. Para evitar que los dispositivos interfieran entre sí, se usan los *buffers* triestado para desconectar todos los dispositivos excepto aquellos que se estén comunicando en cada momento determinado.

Contienda de bus

La contienda de bus tiene lugar cuando dos o más dispositivos tratan de imponer niveles lógicos opuestos en la misma línea de bus común. La forma más habitual de contienda de bus se produce cuando un dispositivo no ha terminado de desconectarse antes de que otro dispositivo conectado al bus empiece a operar. Esto suele ocurrir en los sistemas de memoria al conmutar del modo de lectura al modo de escritura, o viceversa, y aparece como resultado algún problema de temporización.

Líneas de E/S multiplexadas

Algunos dispositivos que envían y reciben datos tienen líneas de entrada y salida combinadas, denominadas puertos de E/S, que es preciso multiplexar en el bus de datos. Para conectar el dispositivo con el bus se utilizan *buffers* triestado bidireccionales, como se ilustra en la Figura 12.34(a).

Cada puerto de E/S tiene un par de *buffers* triestado. Cuando la línea $\overline{SND}/RCV$ ($\overline{\text{Envío}}$/Recepción) está a nivel BAJO, se habilita el *buffer* triestado superior de cada pareja, mientras que el inferior se inhabilita. En

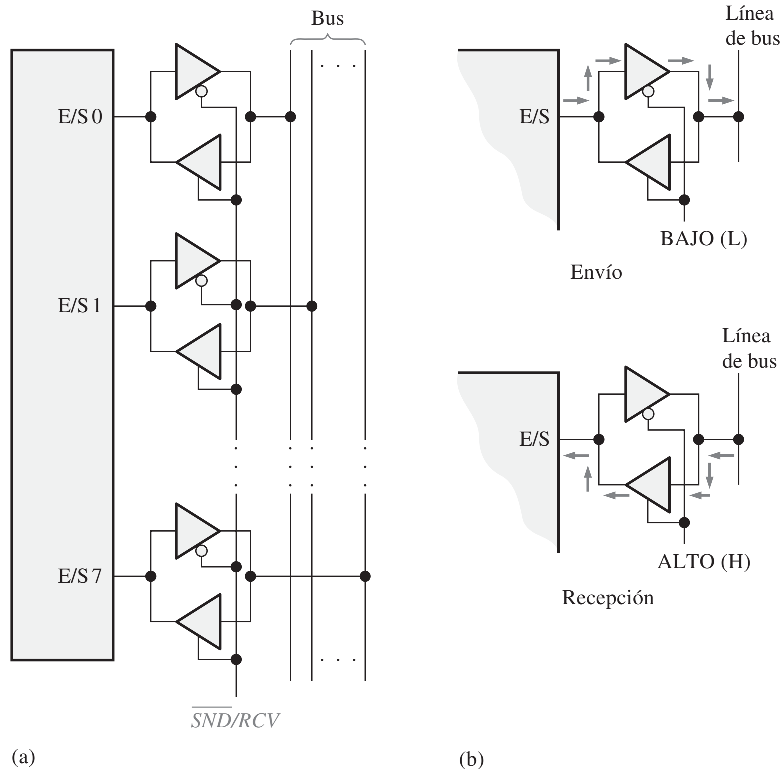


FIGURA 12.34 Operación de E/S multiplexada.

este estado, el dispositivo actúa como origen y envía datos hacia el bus. Cuando la línea $\overline{SND}/RCV$ está a nivel ALTO, se habilita el *buffer* triestado inferior de cada pareja, de modo que el dispositivo actúa como un aceptor y recibe datos del bus. Esta operación se ilustra en la Figura 12.34(b). Algunos dispositivos implementan las operaciones de E/S multiplexadas mediante circuitería interna.

REVISIÓN DE LA SECCIÓN 12.7

1. ¿Por qué se necesitan *buffers* triestado para conectar dispositivos digitales a un bus?
2. ¿Cuál es el propósito de un sistema de bus?

12.8 BUSES ESTÁNDAR

Un bus es una "autopista" para las señales digitales; consiste en un conjunto de conexiones físicas (pistas de circuito impreso o cables), por las que se desplazan los datos y otras informaciones desde un lugar a otro. Un bus también consiste en un conjunto estándar de especificaciones que indican las características y tipos de las señales que pueden viajar a su través. Los buses internos interconectan los distintos componentes dentro de un sistema informático: procesador, memoria, unidad de disco, tarjetas controladoras y tarjetas de interfaz. Los buses externos o de E/S permiten transferir señales digitales entre una computadora y el "mundo exterior" y constituyen la interfaz de la computadora con equipos periféricos (monitor de vídeo, teclado, ratón e impresora) o con otros equipos que deban ser controlados mediante una computadora, como puedan ser instrumentos de prueba y medida.

Al finalizar está sección, el lector deberá ser capaz de:

■ Describir los diversos tipos de buses internos y externos, tanto serie como paralelo. ■ Definir el concepto de bus local. ■ Describir los estándares PCI e ISA de bus interno. ■ Decribir el bus RS-232. ■ Describir la interfaz FireWire. ■ Describir la interfaz USB. ■ Explicar la interfaz GPIB. ■ Describir la interfaz SCSI.

Buses internos

Los buses internos de una computadora transmiten direcciones, datos y señales de control entre el microprocesador, la memoria caché, la memoria SRAM, la memoria DRAM, las unidades de disco, las ranuras de expansión y otros dispositivos internos. La mayor parte de las computadoras personales de hoy en día poseen tres tipos de buses internos: el *bus local,* el *bus PCI* y el *bus ISA*. La Figura 12.35 muestra la disposición básica de los buses en un sistema.

Bus local. Este bus conecta directamente el microprocesador a la memoria caché, a la memoria principal, al coprocesador y al controlador de bus PCI. El **bus local** es el único bus interno que se conecta directamente al microprocesador. Generalmente, este bus incluye los buses de datos, de direcciones y de control que permiten al microprocesador comunicarse con los otros dispositivos. El bus local puede considerarse como el bus *principal* en un sistema informático. Por ejemplo, el bus local del Pentium consta del bus de direcciones con 32 líneas de dirección de memoria, el bus de datos con 64 líneas de datos y el bus de control con numerosas líneas de control.

Bus PCI (Peripheral Control Interconnect, interconexión de control de periféricos). Este bus sirve para establecer la interfaz entre el microprocesador y una serie de dispositivos externos a través de ranuras de expansión (conectores). El **bus PCI** fue desarrollado por Intel y, desde que fue introducido en 1993, se ha convertido en el bus de interfaz estándar para las computadoras personales, desplazando a diversos estándares de bus más antiguos. PCI es un bus de 64 bits, aunque a menudo se lo implementa como un bus de 32 bits en el que

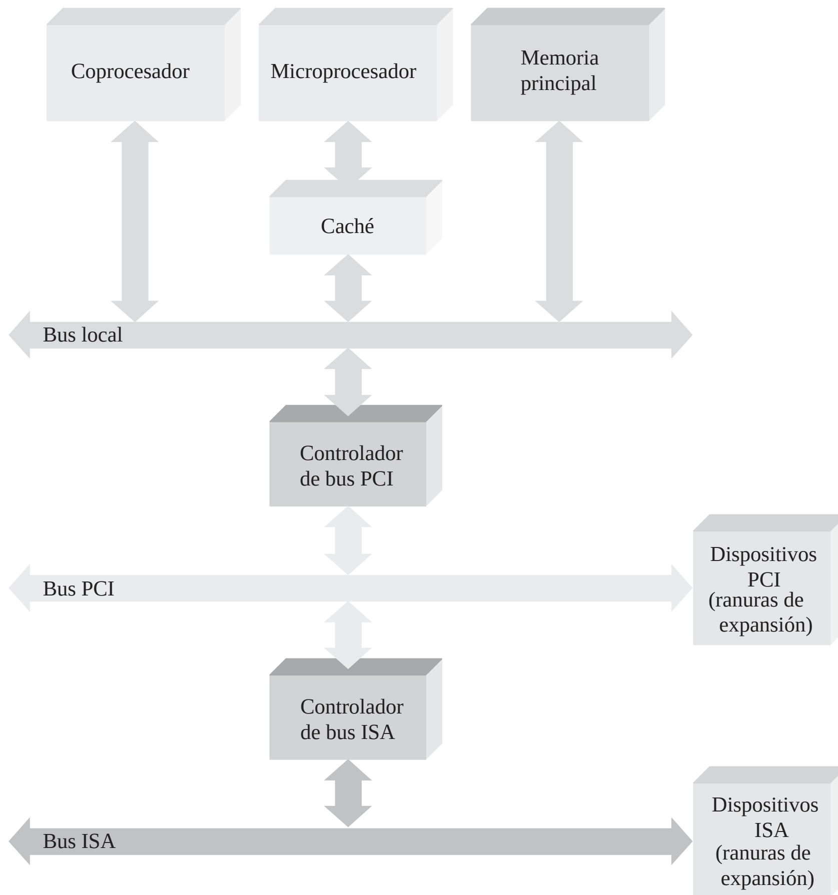


FIGURA 12.35 Ilustración simplificada del sistema básico de buses en una computadora personal típica.

los buses de direcciones y de datos están multiplexados. Puede operar a velocidades de reloj de 33 MHz ó 66 MHz.

El bus PCI está aislado del bus local mediante una unidad controladora de bus que actúa como "puente" entre los dos buses. El PCI se considera un bus *secundario* y su reloj es independiente del reloj del microprocesador. El PCI puede conectar el microprocesador con dispositivos periféricos, como puede ser un disco duro, a través de tarjetas adaptadoras insertadas en ranuras de expansión.

La interfaz PCI es compatible con las características "*plug-and-play*", que son la capacidad que tienen ciertas computadoras para configurar de forma automática las tarjetas de expansión y otros dispositivos. Esto permite conectar un dispositivo a una computadora sin preocuparse de configurar conmutadores, cambiar los puentes ni manejar ningún otro elemento de configuración. Esto se consigue mediante una memoria de 256 bytes que permite a la computadora interrogar a la interfaz PCI.

Bus ISA (Industry Standard Architecture. arquitectura estándar de la industria). Este bus de expansión fue desarrollado por IBM para su computadora personal AT y es el bus estándar en el que se insertan casi todas las tarjetas de circuito impreso fabricadas antes de 1993. La interfaz **ISA** se suele incorporar en las computadoras personales más recientes como complemento del bus PCI, para propósitos de compatibilidad descendente.

La interfaz ISA tiene un bus de datos de 8 o 16 bits y puede operar a 8,33 MHz. Una versión ampliada denominada EISA proporciona un bus de datos de 32 bits, pero se ha dejado prácticamente de utilizar, debido a su baja velocidad, habiendo sido remplazada por el bus PCI.

Buses externos

Los dispositivos externos se conectan a una computadora mediante una interfaz de entrada/salida (E/S) denominada *puerto*. Existen dos tipos básicos de puertos en una computadora, el *puerto serie* y el *puerto paralelo*, y la mayoría de las computadoras tienen un puerto paralelo y, al menos, un puerto serie para conectar modems, impresoras, ratones y otros dispositivos periféricos.

Un puerto serie se usa para la comunicación de datos serie, donde sólo se transfiere un bit cada vez. Los modems y los ratones son ejemplos de dispositivos serie típicos. Algunas veces, los puertos serie también se emplean para conectar la computadora con los equipos de medida y pruebas. Un puerto paralelo se utiliza para la comunicación de datos en paralelo, en la que al menos 1 byte (8 bits) se transfieren cada vez. Actualmente, existen varios estándares de bus en uso, tanto para los puertos serie como paralelo. A continuación se describen los más destacables.

Buses serie para interfaz de E/S

RS-232C Aprobado por la asociación EIA (*Electronic Industries Association*) es uno de los estándares más antiguos y más comunes para establecer interfaces serie. El estándar RS-232C también se denomina EIA-232. La mayoría de los ***modems*** (*mod*ulador/*dem*odulador) cumplen el estándar EIA-232, y la mayor parte de las computadoras personales disponen de un puerto RS-232C. El ratón y algunos monitores e impresoras serie, además de los modems, se diseñan para conectarse al puerto RS-232C. El estándar RS-232C se usa habitualmente para establecer la interfaz entre un equipo terminal de datos, **DTE** (*Data Terminal Equipment*) y un equipo de comunicación de datos, **DCE** (*Data Communication Equipment*). Por ejemplo, una computadora se clasifica como un DTE y un modem como un DCE.

El estándar EIA-232 especifica veinticinco líneas de conexión entre un DTE y un DCE que requieren un conector de veinticinco pines (DB-25), como se muestra en la Figura 12.36. En las aplicaciones de computadoras personales, no se requieren todas las señales RS-232C. Normalmente, se emplean un mínimo de tres y un máximo de once. Por esta razón, IBM definió un conector de 9 pines (DB-9) para su interfaz serie.

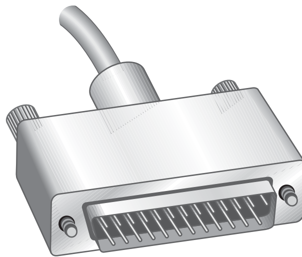
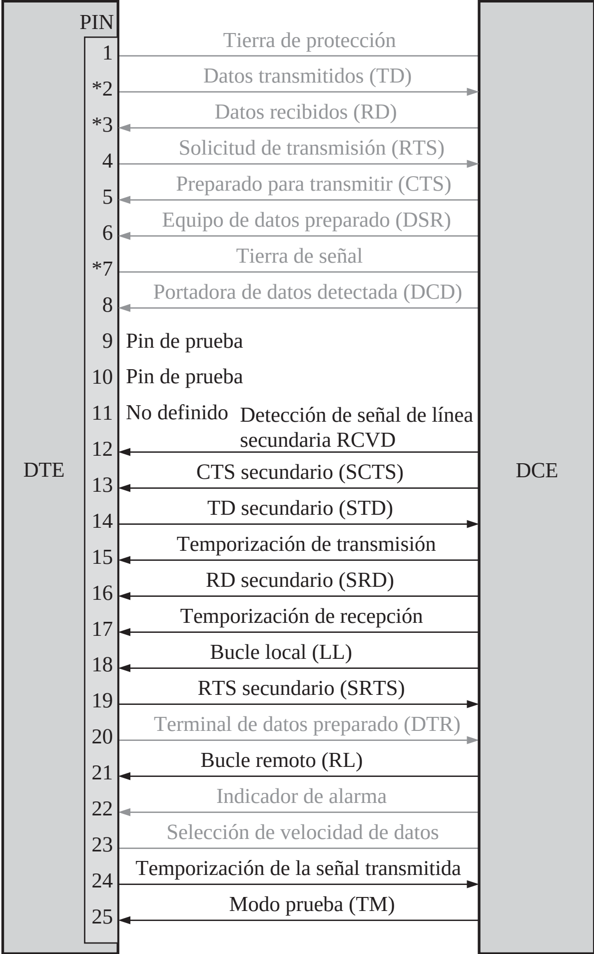
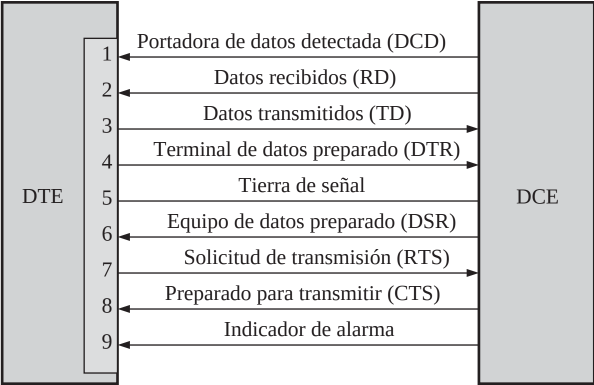


FIGURA 12.36 Conector de 25 pines RS-232C.

La Figura 12.37(a) enumera las señales y la asignación de pines para el conector RS-232C de 25 pines y, en la parte (b) de la figura, se enumeran las señales y la asignación de pines correspondientes al conector de 9 pines. Los once pines y señales marcadas en color gris claro en la parte (a) indican las señales típicamente utilizadas en las aplicaciones de computadoras personales. Las tres señales mínimas se han marcado mediante un asterisco (pines 2, 3 y 7).



(a) Interfaz de 25 pines completa RS-232C con la configuración típica de computadora personal indicada en gris claro y tres señales marcadas con asterisco (pines 2, 3 y 7)



(b) Interfaz RS-232C de 9 pines

FIGURA 12.37 Asignación de pines y señales para ambas versiones de conector RS-232C.

La longitud máxima de cable especificada para la conexión RS-232C es de aproximadamente 150 metros, con una velocidad de transferencia de datos de 20 kbaudios. Si se utiliza un cable más corto, la velocidad en baudios puede ser mayor. La especificación de la velocidad de transferencia de datos en baudios y en bits por segundo (bps) no tienen necesariamente por qué coincidir. La razón de ello es que la velocidad en baudios corresponde a la terminología empleada para los modems y se define como el número de cambios de la señal por segundo, lo que se denomina velocidad de modulación. En los modems, algunas veces un cambio de señal transfiere varios bits de datos. Para velocidades bajas, la tasa de transferencias en baudios es igual a la medida de bits por segundo pero, a velocidades altas, la tasa de transferencia en baudios puede ser menor que dicha medida.

Para superar las limitaciones de la interfaz RS-232C, se desarrollaron otros dos estándares: el RS-422 y el RS-423. Estos nuevos estándares especifican longitudes de cable mucho más largas y velocidades de transferencia de datos más altas bajo determinadas condiciones. Por ejemplo, tanto el estándar RS-422 como el RS-423 especifican una longitud máxima de cable de aproximadamente 1.200 metros. La velocidad de transferencia de datos máxima RS-422 es de 10 Mbaudios para un cable de 12 metros y 100 kbaudios para 1.200 metros. Para el estándar RS-423, la velocidad de transferencia de datos es de 100 kbaudios para 90 metros y 1 kbaudio para 1.200 metros. La interfaz RS-232C continúa actualmente siendo la más común.

IEEE 1394. Es un estándar de bus serie externo relativamente nuevo y muy rápido, que admite velocidades de transferencia de hasta 400 Mbps y, típicamente, aunque no exclusivamente, se emplea para conectar con los periféricos de vídeo y gráficos, como las cámaras digitales. El estándar **IEEE 1394** a menudo se denomina ***FireWire***, un nombre registrado por Apple, que fue quien primero lo desarrolló. Otras empresas utilizan otros nombres para describir sus productos IEEE 1394. IEEE son las siglas de *Institute of Electrical and Electronics Engineers* (Instituto de ingenieros eléctricos y electrónicos).

Se pueden conectar hasta 64 dispositivos a un bus FireWire, con una tecnología de cadena (*daisy chain*). El cable FireWire consta de seis conductores, dos pares trenzados para datos y dos para la alimentación. Este estándar también admite la posibilidad de "conexión en caliente", es decir, la capacidad de añadir o eliminar de dispositivos conectados a una computadora mientras que ésta está en funcionamiento.

USB (Universal Serial Bus, bus serie universal). El **USB** proporciona dos velocidades de transferencia de datos, una alta velocidad de 12 Mb/s y una baja velocidad de 1,5 Mb/s. Se puede emplear un puerto USB para conectar hasta 127 dispositivos periféricos y el estándar admite tanto características *plug-and-play* como de conexión en caliente. El cable USB consta de cuatro conductores, dos para datos y dos para alimentación, y conecta la computadora a los dispositivos periféricos USB, pudiendo actuar cualquiera de ellos como concentrador (*hub*) para establecer la conexión con otros dispositivos periféricos USB. La Figura 12.38 ilustra un sistema de computadora con interfaz USB.

Buses paralelo para interfaz de E/S

IEEE 488 Este estándar de bus se emplea hace tiempo y se conoce también con el nombre de bus de interfaz de propósito general (**GPIB,** *General Purpose Interface Bus*). Ampliamente utilizado en aplicaciones de medida y pruebas, fue desarrollado por Hewlett-Packard en los años sesenta. El estándar **IEEE 488** especifica 24 líneas, que se usan para transferir ocho bits de datos en paralelo a la vez y proporcionar ocho señales de control, que incluyen tres líneas para el establecimiento de la comunicación y cinco líneas para el gobierno del bus. También incluye ocho líneas de tierra para apantallamiento y para los retornos a masa. La velocidad de transferencia de datos máxima para el estándar IEEE 488 es de 1 MB/s. Un superconjunto de este estándar, denominado HS488, proporciona una velocidad de transferencia de datos máxima de 8 MB/s.

Para conectar un equipo de pruebas a una computadora utilizando el bus IEEE 488, se instala una tarjeta de interfaz en la computadora, que convierte a la computadora en un **controlador** del sistema. En una configuración GPIB típica, se pueden conectar al controlador del sistema hasta 14 dispositivos (instrumentos de

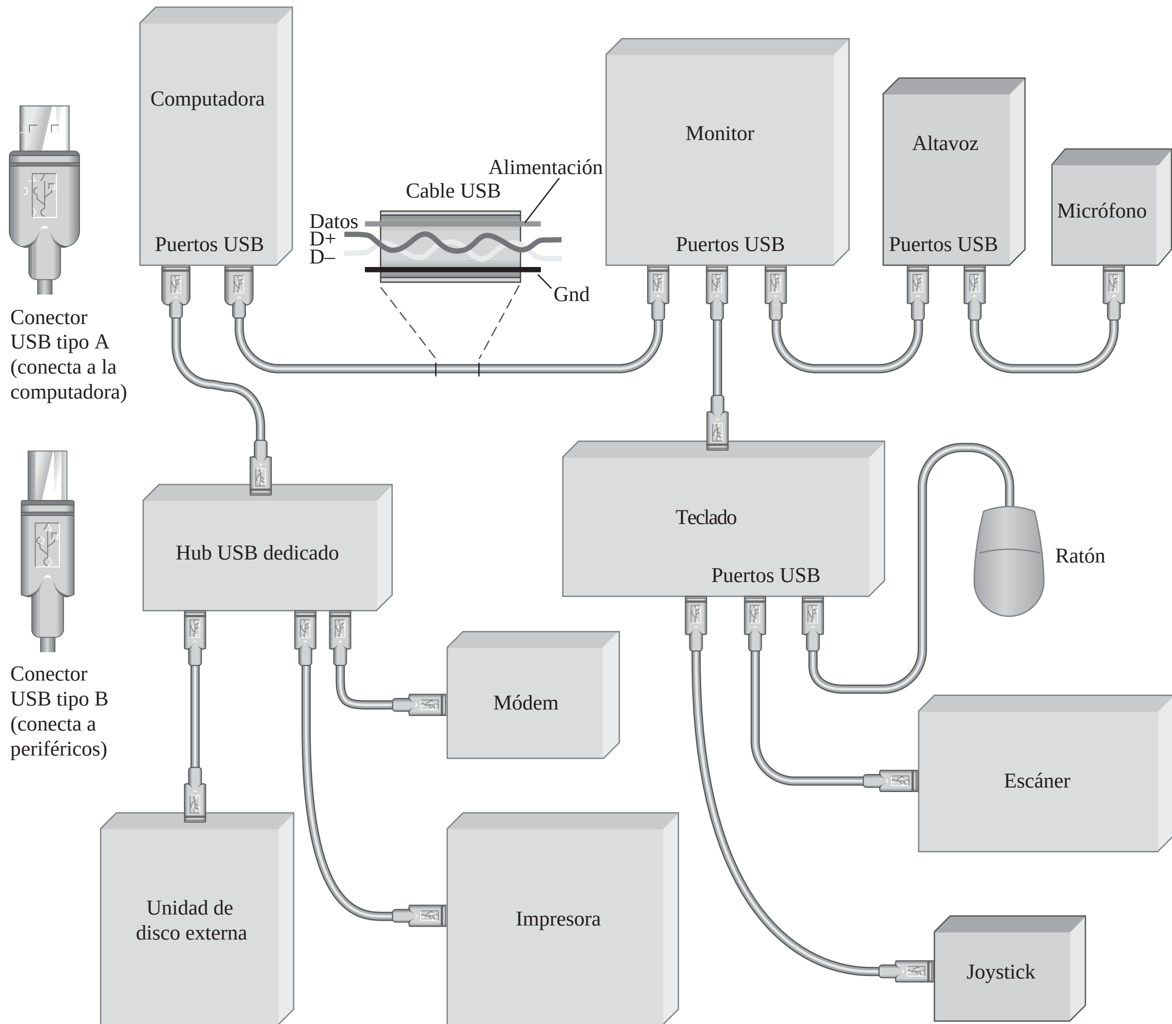


FIGURA 12.38 Ejemplo de un sistema de computadora con interfaz USB.

medida y pruebas). Cuando el controlador del sistema envía un comando dirigido a un dispositivo controlado, con el fin de que éste lleve a cabo una operación específica, como por ejemplo una medida de frecuencia, se dice que el controlador "habla" (transmisor) y que el dispositivo controlado "escucha" (receptor).

Un **receptor** es un instrumento capaz de recibir datos en una configuración GPIB cuando el controlador del sistema (computadora) se dirige a él. Ejemplos de escuchas son las impresoras, monitores, fuentes de alimentación programables y generadores de señal programables. Un **transmisor** es un instrumento capaz de enviar datos a través del bus GPIB. Ejemplos de transmisores son los multímetros digitales y los contadores de frecuencia que pueden generar datos compatibles con el bus. Algunos instrumentos pueden enviar y recibir datos y se denominan transmisores/receptores; ejemplos de ellos son las computadoras, modems y ciertos instrumentos de medida. El controlador del sistema puede definir a cada uno de los otros instrumentos que hay conectados al bus como receptores o transmisores, de cara a la transferencia de datos. Normalmente, el controlador es un transmisor/receptor.

En la Figura 12.39 se presenta, como ejemplo, una disposición típica GPIB. Las tres agrupaciones básicas de señal de bus se muestran como *bus de datos, bus de control de transferencia de datos* y *bus de gobierno de la interfaz.*

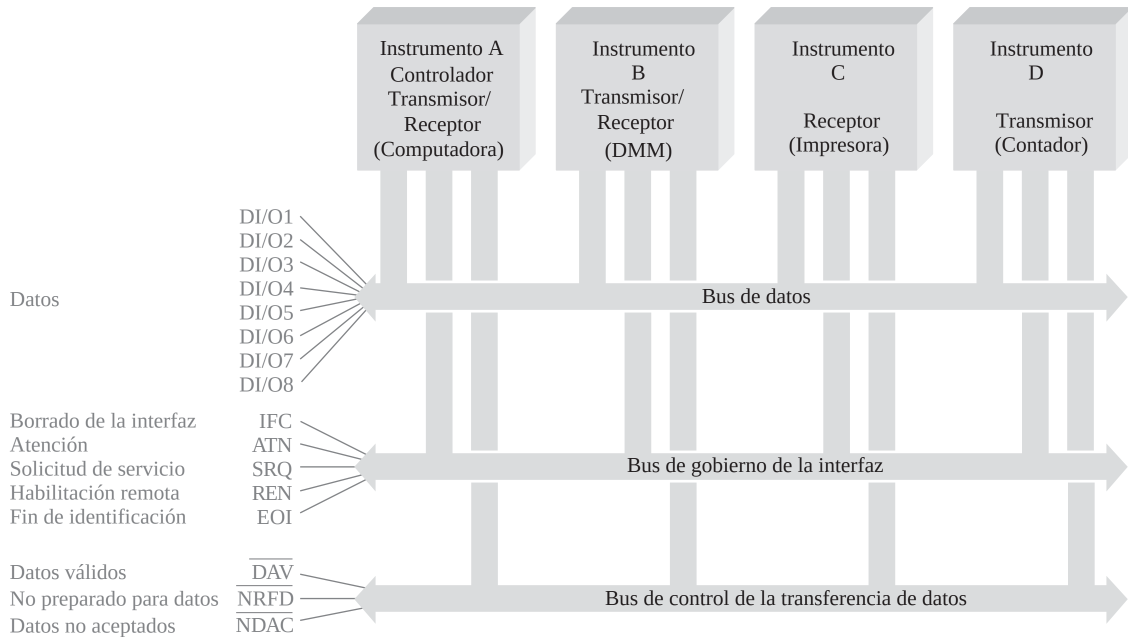


FIGURA 12.39 Una conexión típica IEEE 488 (GPIB)

Las líneas de datos en paralelo se designan como DI/O1a DI/O8 (entrada/salida de datos). En esta parte bidireccional del bus se transfiere un byte de datos. Cada byte que se transfiere requiere una secuencia de negociación a través del bus de transferencia. Las tres líneas para la negociación, activas a nivel BAJO, indican si los datos son válidos $(\overline{DAV})$, si el instrumento direccionado no está preparado para los datos $(\overline{NFRD})$ o si los datos no son aceptados $(\overline{NDAC})$. Más de un instrumento puede recibir datos a un mismo tiempo, y es el instrumento más lento el que establece la velocidad de transferencia. La Figura 12.40 muestra el diagrama de tiempos para la secuencia de establecimiento de comunicación GPIB y la Tabla 12.2 describe las señales para el establecimiento de la comunicación.

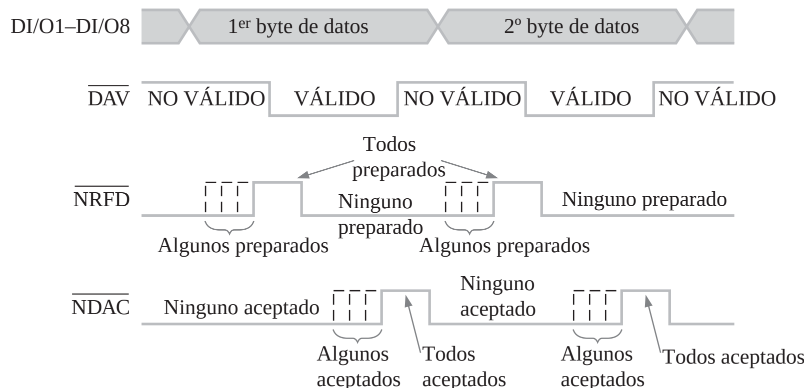


FIGURA 12.40 Cronograma de la secuencia de establecimiento de la comunicación GPIB.

Las cinco señales del bus de gobierno de la interfaz controlan el flujo ordenado de los datos. La línea ATN (*attention*) es supervisada por todos los instrumentos conectados al bus. Cuando la línea ATN está activa, el controlador selecciona la operación de interfaz específica, designa a los transmisores y receptores y proporciona un direccionamiento específico para los receptores. Cada instrumento diseñado según el estándar GPIB tiene una dirección de identificación específica, usada por el controlador del sistema. La Tabla 12.3 describe las líneas de gobierno de la interfaz GPIB y sus funciones.

Nombre	Descripción
$\overline{\text{DAV}}$	**Data Valid:** después de que el transmisor detecta un nivel alto en la línea $\overline{\text{NFRD}}$, pone esta línea a nivel BAJO cuando los datos en sus líneas de E/S han sido establecidos y son válidos.
$\overline{\text{NRFD}}$	**Not Ready for Data:** el receptor pone esta línea a nivel bajo para indicar que no está preparado para recibir datos. Un nivel ALTO indica que sí está preparado. La línea $\overline{\text{NRFD}}$ no pasará a nivel ALTO hasta que todos los receptores direccionados estén preparados para recibir datos.
$\overline{\text{NDAC}}$	**Not Data Accepted:** el receptor pone esta línea a nivel BAJO para indicar que no ha aceptado los datos. Cuando acepta datos de sus líneas de E/S, desactiva su línea $\overline{\text{NDAC}}$. La línea $\overline{\text{NDAC}}$ de entrada al transmisor no pasa a nivel ALTO hasta que todos los receptores han aceptado los datos.

TABLA 12.2 Señales para el establecimiento de la comunicación del GPIB.

Nombre	Descripción
ATN	**Attention:** hace que todos los dispositivos conectados al bus interpreten los datos como una dirección o un comando del controlador, y activa la función de establecimiento de la comunicación.
IFC	**Interface Clear:** inicializa el bus
SRQ	**Service Request:** alerta al controlador de que un dispositivo necesita comunicarse.
REN	**Remote Enable:** habilita a los dispositivos para responder a un control remoto de programación.
EOI	**End or Identify:** indica el último byte de datos que se transfiere.

TABLA 12.3 Líneas de gobierno del bus GPIB.

El GPIB está limitado a una longitud máxima de cable de 15 metros y no puede haber más de un dispositivo por metro, con una capacidad de carga máxima de 50 pF por dispositivo. La limitación de la longitud de cable se puede superar utilizando modems y extensores de bus. Un extensor de bus permite conectar el cable de interfaz a los instrumentos que están separados por una distancia mayor que la permitida por el GPIB, o comunicarse a larga distancia vía modem-línea telefónica. En la Figura 12.41 se ilustra el uso de los extensores de bus y/o modems.

SCSI (Small Computer System Interface, interfaz para sistemas informáticos de pequeño tamaño). Se trata de un estándar ampliamente utilizado para conectar computadoras personales y periféricos. Aunque ***SCSI*** es un estándar ANSI (*American National Standards Institute,* Instituto nacional de estándares de EE.UU.), existen diversas variaciones y tipos de conectores, de distintos fabricantes. Puede ocurrir que un tipo de SCSI no sea compatible con otro tipo. SCSI-1 es la versión con conector de 25 pines que proporciona un bus de datos de 8 bits

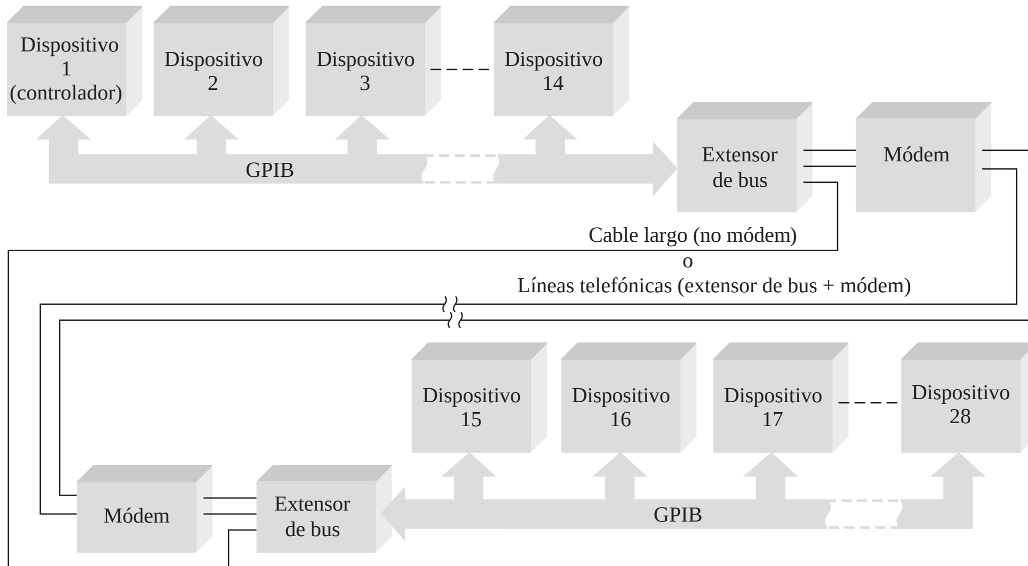


FIGURA 12.41 Se puede utilizar un extensor de bus y modem para establecer la interfaz con sistemas GPIB remotos.

y admite velocidades de transferencia de datos de 4 MB/s. A continuación se enumeran otras versiones del estándar de bus SCSI:

- *SCSI-2*. Esta versión es la misma que SCSI-1, pero utiliza un conector de 50 pines y admite múltiples dispositivos.
- *Wide SCSI* (SCSI ancho). Utiliza un conector más ancho que el tipo SCSI-2, para dar soporte a transferencias de datos de 16 bits.
- *Fast SCSI* (SCSI rápido). Proporciona transferencia de datos de 8 bits, pero admite velocidades de transferencia de datos de 10 MB/s.
- *Fast Wide SCSI*. Esta versión permite la transferencia de datos de 16 bits a 20 MB/s.
- *Ultra SCSI.* Esta versión transfiere 8 bits de datos a 20 MB/s.
- *SCSI*-3. Esta versión tiene 16 líneas de datos y funciona a 40 MB/s.
- *Ultra SCSI*-2. Esta versión transfiere 8 bits a 40 MB/s.
- *Wide Ultra SCSI*-2. Esta versión proporciona transferencias de datos de 16 bits y opera a 80 MB/s.

Las descripciones de señal para un conector de 25 pines SCSI se proporcionan en la Tabla 12.4, mientras que la configuración de pines se muestra en la Figura 12.42.

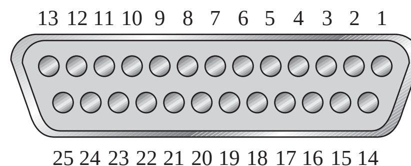


FIGURA 12.42 Conector de 25 pines SCSI.

Número de pin	Nombre de la señal	Descripción de la señal
1	REQ/	Request (Solicitud)
2	MSG/	Message (Mensaje)
3	I/O/	Input/Output (Entrada/salida)
4	RST/	SCSI bus reset (Puesta a cero del bus SCSI)
5	ACK/	Acknowledge (Confirmación)
6	BSY/	Busy (Ocupado)
7	GND	Señal de tierra
8	DB0/	Bit de datos 0
9	GND	Señal de tierra
10	DB3/	Bit de datos 3
11	DB5/	Bit de datos 5
12	DB6/	Bit de datos 6
13	DB7/	Bit de datos 7
14	GND	Señal de tierra
15	C/D/	Command/Data (Comando/datos)
16	GND	Señal de tierra
17	ATN/	Attention (Atención)
18	GND	Señal de tierra
19	SEL/	Select (Selección)
20	DBP/	Paridad de datos
21	DB1/	Bit de datos 1
22	DB2/	Bit de datos 2
23	DB4/	Bit de datos 4
24	GND	Señal de tierra
25	TPWR	Alimentación del terminador

TABLA 12.4 Señales SCSI.

REVISIÓN DE LA SECCIÓN 12.8

1. Nombrar las dos principales categorías de buses, en términos del método de transferencia de datos.
2. Clasificar cada uno de los siguientes buses como bus serie o bus paralelo:
   (a) SCSI **(b)** RS-232C **(c)** USB **(d)** GPIB
3. Explicar la diferencia básica entre un bus serie y un bus paralelo.
4. ¿Cuántos dispositivos pueden conectarse al bus USB?
5. ¿Es el bus FireWire más rápido que el USB, en términos de transferencia de datos?

RESUMEN

- Las unidades básicas de una computadora se muestran en la Figura 12.43.

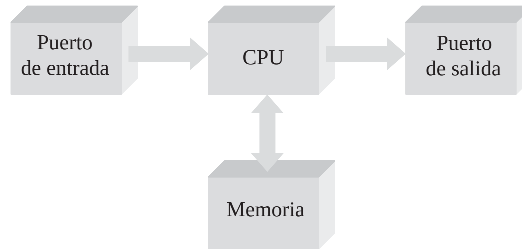


FIGURA 12.43

- Los tres buses básicos de una computadora son el *bus de direcciones*, el *bus de datos* y el *bus de control*. El tamaño de cualquiera de estos buses viene especificado por el número de hilos conductores distintos.
- Entre los dispositivos periféricos típicos se incluyen el teclado, las unidades de disco externas, el ratón, la impresora, el módem y el escáner.
- El número de líneas de dirección varía desde 20, en el 8086 /8088, hasta 32 para la familia de procesadores Pentium. Originalmente, el bus de datos era de 16 bits para el 8086 y ahora es de 64 para la familia Pentium.
- Los registros de propósito general son un subconjunto de los existentes en todos los procesadores Intel. Dichos registros incluyen:

  Acumulador (AX, que incluye AH y AL)
  Base (BX, que incluye BH y BL)
  Contador (CX, que incluye CH y CL)
  Datos (DX, que incluye DH y DL)
  Puntero de pila (SP)
  Puntero de base (BP)
  Índice de origen (SI)
  Índice de destino (DI)

  A partir del 80386, este conjunto básico se amplió al conjunto de registros ampliado.
- Los registros de segmento básicos son un subconjunto de los registros existentes en todos los procesadores Intel. Los registros de segmento son:

  Segmento de código (CS)
  Segmento de datos (DS)
  Segmento de pila (SS)
  Segmento extra (ES)

  A partir del 80386, se añadieron dos nuevos registros de segmento.
- Los registros indicadores (flag) son un subconjunto de los registros existentes en todos los procesadores Intel. Los registros indicadores incluyen:

  Captura (TF)
  Dirección (DF)
  Habilitación de interrupción (IF)
  Desbordamiento (OF)
  Signo (SF)
  Cero (ZF)
  Acarreo auxiliar (AF)
  Paridad (PF)
  Acarreo (CF)

- El lenguaje "básico" de una computadora se denomina código máquina y sus instrucciones se especifican como una serie de códigos binarios.
- En el lenguaje ensamblador, las instrucciones en código máquina se reemplazan por mnemónicos alfabéticos abreviados que tienen una correspondencia directa con el código máquina. El lenguaje ensamblador también utiliza directivas que permiten al programador especificar otros parámetros que no se traducen directamente a código máquina.
- Los puertos constituyen una interfaz con los dispositivos externos. Se pueden configurar como puertos de entrada, de salida o una combinación de ambos. Puede accederse a ellos como puertos dedicados o asignados a memoria y se les puede dar servicio mediante sondeo, mediante interrupciones o mediante software.
- La Tabla 12.5 muestra una comparativa de los buses estándar.

	Buses internos		**Buses externos**				
	PCI	**ISA**	**RS-232C**	**IEEE 1394**	**USB**	**IEEE 488**	**SCSI**
Tipo	Paralelo	Paralelo	Serie	Serie	Serie	Paralelo	Paralelo
Líneas de datos	32/64	8/16	–	–	–	8	8/16
Velocidad de los datos	33/66 MHz	8,33MHz	20 kbaudios	400 Mb/s	1,5/12 Mb/s	1 Mb/s	4 Mb/s (1) 10 Mb/s (Fast) 20 Mb/s (Ultra) 40 Mb/s (3) 80 Mb/s (Ultrawide 2)
Número de dispositivos	–	–	1	63	127	14	16

TABLA 12.5

PALABRAS CLAVE

Las palabras clave y otros términos que se han resaltado en negrita se encuentran en el glosario final del libro.

Bus de control Un conjunto de conductores de una sola dirección que conecta la CPU con otras partes de la computadora, para coordinar sus operaciones y comunicarse con dispositivos externos.

Bus de datos Un conjunto bidireccional de conductores por los que los datos o códigos de instrucciones se transfieren al microprocesador o por los que se envía, desde el microprocesador, el resultado de una operación o cálculo.

Bus de direcciones Generalmente, un grupo de conductores de una sola dirección que va desde el microprocesador a la memoria y que contiene la información de direcciones.

CPU *Central Processing Unit*, unidad central de proceso. El "cerebro" de la computadora que procesa las instrucciones de los programas.

FireWire El bus serie estándar IEEE-1394.

GPIB *General Purpose Interface Bu*s, bus de interfaz de propósito general, basado en el estándar IEEE 488.

Interrupción Señal o instrucción que hace que el proceso actual sea temporalmente detenido mientras se ejecuta una rutina de servicio.

Lenguaje de alto nivel Un tipo de lenguaje de computadora muy próximo al lenguaje humano que se encuentra en un nivel por encima del lenguaje ensamblador.

Lenguaje ensamblador Un tipo de lenguaje de programación que utiliza palabras similares a las del inglés y que tiene una correspondencia biunívoca con lenguaje máquina.

Lenguaje máquina Instrucciones de computadora escritas en código binario que una computadora es capaz de comprender. El nivel más bajo de lenguaje de programación.
Microprocesador Circuito integrado digital de muy gran escala que puede ser programado para realizar operaciones aritméticas, lógicas u otras operaciones. La CPU de una computadora.
Módem Modulador/demodulador que sirve para conectar dispositivos digitales con sistemas de transmisión analógicos, como las líneas telefónicas.
Periférico Dispositivo como por ejemplo una impresora o un módem que proporciona servicios de comunicación con una computadora.
Programa Lista de instrucciones que una computadora sigue para alcanzar un resultado específico.
Puerto Una interfaz física en una computadora, a través de la cual se transfieren datos hacia o desde un periférico.
SCSI *Small Computer System Interface* (interfaz para sistemas informáticos de pequeño tamaño). Estándar de bus paralelo externo.
Triestado Un tipo de salida en un circuito lógico que posee tres estados: ALTO, BAJO y alta-Z. Se emplea para establecer la interfaz de las salidas de un dispositivo de origen con un bus.
USB *Universal Serial Bus* (bus serie universal). Estándar de bus serie externo.

AUTOTEST

Las respuestas se encuentran al final del capítulo.

1. Una computadora básica no incluye:
(a) unidad aritmético-lógica **(b)** unidad de control
(c) unidades periféricas **(d)** unidad de memoria

2. Un bus de direcciones de 20 bits admite:
(a) 100.000 direcciones de memoria **(b)** 1.048.576 direcciones de memoria
(c) 2.097.152 direcciones de memoria **(d)** 20.000 direcciones de memoria

3. El número de bits en el bus de datos del procesador Pentium es:
(a) 16 **(b)** 24 **(c)** 32 **(d)** 64

4. Un bus que se usa para transferir información desde y hacia el microprocesador es el:
(a) bus de direcciones **(b)** bus de datos
(c) los dos anteriores **(d)** ninguno de los anteriores

5 Un ejemplo de unidad periférica es:
(a) un registro de dirección **(b)** la MPU
(c) el monitor de vídeo **(d)** el adaptador de interfaz

6. Los dos tipos de transferencias de memoria que emplea la CPU son:
(a) directa y por interrupción **(b)** lectura y escritura
(c) por bus y multiplexada **(d)** entrada y salida

7. En la familia Intel, el número máximo de dispositivos de E/S de 8 bits es:
(a) 64 **(b)** 1000 **(c)** 64.000
(d) 1 millón **(e)** ilimitado

8. El sondeo es un método utilizado para:
(a) determinar el estado del microprocesador.
(b) establecer la comunicación entre la CPU y un periférico
(c) establecer una prioridad para la comunicación con varios periféricos.
(d) determinar la siguiente instrucción

9. De los siguientes registros, ¿cuál es un registro de 8 bits?
(a) AH **(b)** BX **(c)** SS **(d)** IP

10. Esencialmente, un mnemónico es:
(a) un diagrama de flujo **(b)** un operando
(c) código máquina **(d)** una instrucción

11. DMA quiere decir:
(a) *digital microprocessor address* (dirección de microprocesador digital)
(b) *direct memory access* (acceso directo a memoria)
(c) *data multiplexed access* (acceso multiplexado a datos)
(d) *direct memory addressing* (direccionamiento directo de memoria)

12. Un programa de computadora es una lista de:
(a) direcciones de memoria que contienen los datos que hay que usar en una operación.
(b) direcciones que contienen instrucciones que hay que utilizar en una operación.
(c) instrucciones ordenadas para conseguir un resultado específico.

13. Un tipo de estructura en lenguaje ensamblador que altera el curso del programa se denomina:
(a) bucle **(b)** salto
(c) los dos anteriores **(d)** Ninguno de los anteriores

14. Un tipo de interrupción que se invoca desde dentro de un programa se denomina:
(a) interrupción software **(b)** interrupción por sondeo
(c) interrupción directa **(d)** interrupción de E/S

15. La mayoría de los dispositivos realizan la interfaz con un bus mediante:
(a) salidas totem-pole **(b)** *buffers* triestado
(c) transistores *pnp* **(d)** resistencias

16. El bus PCI está formado por:
(a) 8 o 16 líneas de datos
(b) 32 o 64 líneas de datos
(c) 1 línea de datos serie

17. Los dispositivos que trabajan con un bus GPIB se denominan:
(a) fuente y carga
(b) transmisor y receptor
(c) donador y aceptor

18. RS-232C es:
(a) una interfaz estándar para datos paralelo
(b) una interfaz estándar para datos serie
(c) una mejora de la interfaz IEEE-488
(d) lo mismo que SCSI

19. El bus FireWire es lo mismo que el:
(a) bus IEEE 488 **(b)** USB
(c) IEEE 1394 **(d)** RS-422 **(e)** RS-423

12. El bus USB admite hasta:
(a) 63 dispositivos **(b)** 14 dispositivos
(c) 100 dispositivos **(d)** 127 dispositivos

PROBLEMAS

Las respuestas a los problemas impares se encuentran al final del libro.

SECCIÓN 12.1 Una computadora básica

1. Nombrar los elementos básicos de una computadora.

2. Nombrar dos categorías de software de computadora.

3. ¿Qué es un bus?

4. ¿Qué es un puerto?

SECCIÓN 12.2 Microprocesadores

5. Nombrar los elementos básicos de un microprocesador.

6. Enumerar tres operaciones que pueda realizar un microprocesador.

7. Citar los tres buses de un microprocesador.

8. ¿Cuáles son los siete grupos básicos del conjunto de instrucciones del Pentium?

SECCIÓN 12.3 Una familia específica de microprocesadores

9. ¿Cuáles son los tres pasos básicos que repite cíclicamente un procesador?

10. ¿Qué significa "*pipelining*"?

11. Nombrar los seis registros de segmento.

12. Suponer que el registro de código de segmento contiene el número hexadecimal 0F05 y que el registro de puntero de instrucción contiene el número 0100. ¿Cuál es la dirección física de la siguiente instrucción que hay que ejecutar?

13. Explicar diferencia entre los registros AH, AL, AX y EAX.

14. **(a)** ¿Qué es un indicador (*flag*)?

(b) ¿Para qué dos propósitos se emplean los indicadores?

15. Explicar la ventaja que representa el emparejamiento de instrucciones en el procesador Pentium.

SECCIÓN 12.4 Programación de computadoras

16. ¿Qué es un ensamblador?

17. Dibujar un diagrama de flujo para un programa que suma los números de 1 a 10 y guarda el resultado en una posición de memoria denominada TOTAL.

18. Dibujar un diagrama de flujo que muestre cómo se podría contar el número de bytes de una cadena de caracteres y almacenar el recuento en una posición de memoria denominado RECUENTO. Suponer que la cadena comienza en una posición denominada INICIO y que usa el carácter de espacio ASCII (20 en hexadecimal) para indicar el final. No debe contabilizarse el carácter de espacio.

19. Explicar qué ocurre cuando se ejecuta la instrucción **mov ax,[bx]**.

20. ¿Qué es un compilador?

SECCIÓN 12.5 Interrupciones

21. Comparar la E/S por sondeo y la E/S controlada por interrupción.

22. ¿Qué significa el término *vectorización*?

23. ¿Qué es una interrupción software?

SECCIÓN 12.6 Acceso directo a memoria (DMA)

24. Explicar qué ocurre en una operación de acceso directo a memoria (DMA).

25. ¿Cómo es la CPU utiliza en DMA?

SECCIÓN 12.7 Interfaces internas

26. En una transferencia serie de ocho bits de datos desde un dispositivo de origen a un dispositivo aceptor, se observa la secuencia de establecimiento de comunicación de la Figura 13.65 en las cuatro líneas de un bus genérico. Analizando las relaciones temporales, identificar la función de cada señal e indicar si se originan en el origen o en el aceptor.

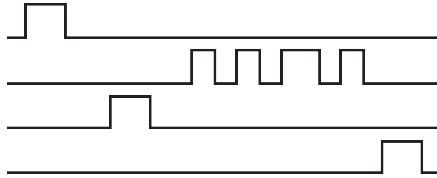


FIGURA 12.44

27. Determinar la señal en la línea de bus de la Figura 12.45 para las formas de onda de habilitación y de entrada de datos mostradas.

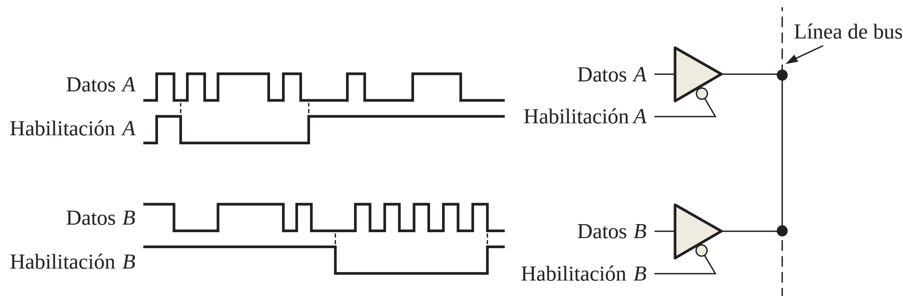


FIGURA 12.45

28. En la Figura 12.46(a), datos procedentes de dos orígenes se colocan en el bus de datos bajo el control de la línea de selección. La señal de selección se muestra en la Figura 12.46(b). Determinar la forma de onda del bus de datos para los códigos de salida de los dispositivos indicados.

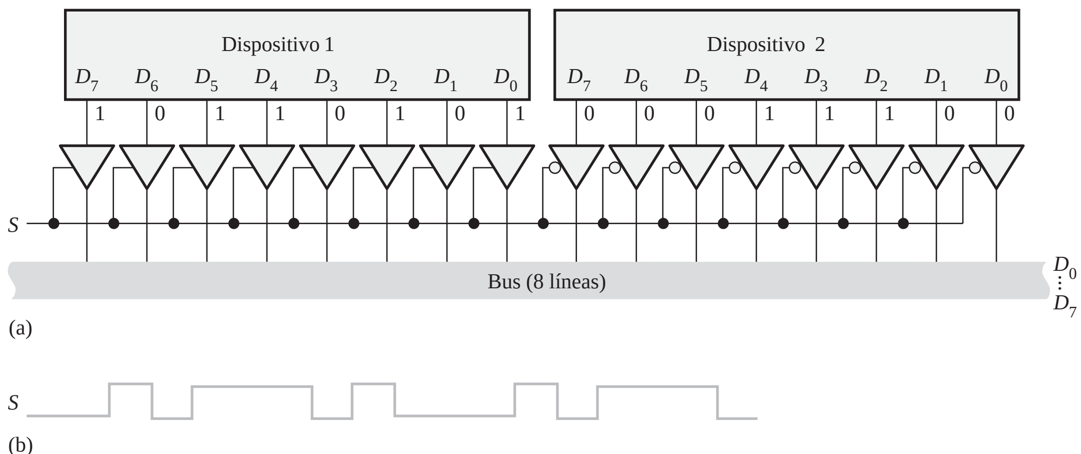


FIGURA 12.46

SECCIÓN 12.8 Buses estándar

29. Explicar la diferencia básica entre un bus local y el bus PCI.

30. Definir "*plug-and-play*".

31. ¿En qué difieren el bus PCI y el bus ISA?

32. Si se emplea una interfaz RS-232C muy corta, ¿pueden transmitirse los datos a una velocidad muy rápida?

33. ¿De qué especificación de bus son parte los DCE y DTE? Indicar qué significan los acrónimos DCE y DTE.

34. Enumerar los hilos de un cable USB.

35. Ocho instrumentos compatibles GPIB se conectan al bus. ¿Cuántos más pueden añadirse sin exceder las especificaciones?

36. Considerar la interfaz GPIB entre un dispositivo transmisor y otro receptor mostrada en la Figura 12.47(a). A partir del cronograma para el establecimiento de la comunicación de la parte (b), determinar cuántos bytes de datos se transfieren realmente al dispositivo receptor.

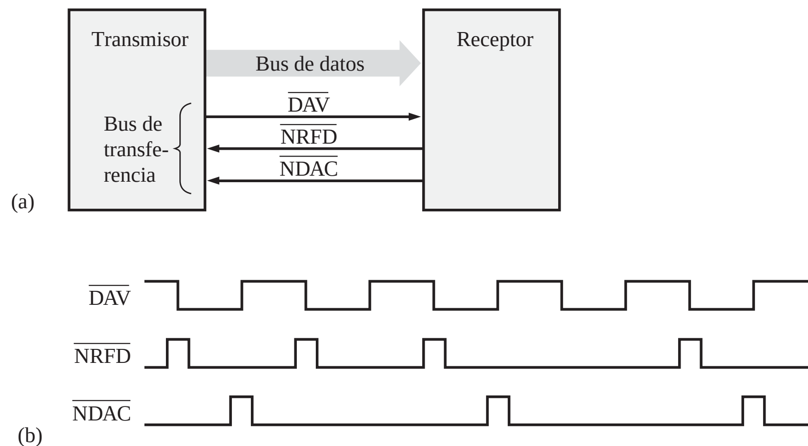


FIGURA 12.47

37. Describir las operaciones mostradas en el cronograma GPIB de la Figura 12.48. Dibujar un diagrama de bloques básico para el sistema implicado en esta operación.

38. Un dispositivo transmisor envía un byte de datos a un dispositivo receptor en un sistema GPIB. Simultáneamente, un DTE envía un byte de datos a un DCE por medio de una interfaz RS-232C. ¿Qué sistema recibirá el primero el byte de datos completo? ¿Por qué?

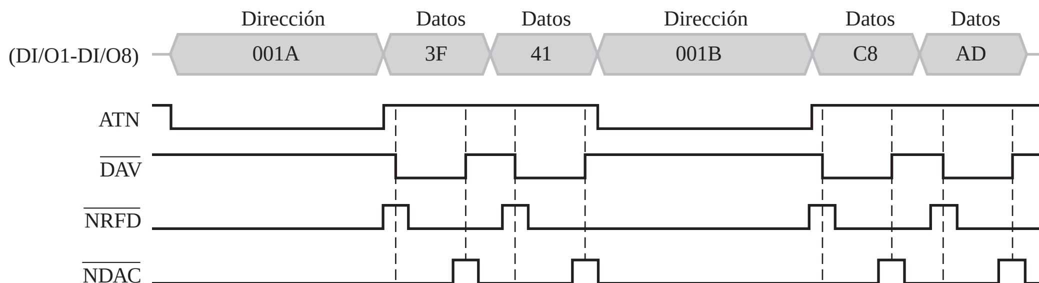


FIGURA 12.48

RESPUESTAS

REVISIÓN DE LAS SECCIONES

SECCIÓN 12.1 Una computadora básica

1. Los elementos básicos de una computadora son la CPU, las memorias, los puertos de entrada/salida y los buses.
2. RAM es una memoria de acceso aleatorio y ROM es una memoria de sólo lectura.
3. Los periféricos son dispositivos externos a la computadora.
4. Hardware es el microprocesador, la memoria, el disco duro, etc. El software es el programa que se ejecuta en la computadora.

SECCIÓN 12.2 Microprocesadores

1. Los elementos de un microprocesador son la unidad aritmético lógica (UAL o ALU), el decodificador de instrucciones, la matriz de registros y la unidad de control.
2. Los buses de un microprocesador son el bus de direcciones, el bus de datos y el bus de control.
3. Un microprocesador funciona como una CPU.
4. Operaciones aritmético/lógicas, movimiento de datos y toma de decisiones.
5. *Pipelining* es el proceso de ejecutar más de una instrucción al mismo tiempo.

SECCIÓN 12.3 Una familia específica de microprocesadores

1. Los registros de propósito general son:

   Acumulador (AX: AH, AL)
   Índice base (BX:BH, BL)
   Contador (CX: CH, CL)
   Puntero de pila (SP)
   Puntero de base (BP)
   Índice de destino (DI)
   Índice de origen (SI)
   Datos (DX: DH, DL)

2. La BIU proporciona una interfaz de datos y direccionamiento.
3. No, la unidad de ejecución no interacciona con los buses.
4. La cola de instrucciones almacena instrucciones pre-extraídas para la unidad de ejecución con el fin de incrementar la velocidad.
5. Los códigos pueden reubicarse fácilmente dentro de la memoria.
6. El emparejamiento de instrucciones es el proceso de combinar instrucciones independientes de modo que puedan ser ejecutadas simultáneamente por las dos unidades de ejecución de un Pentium.

SECCIÓN 12.4 Programación de computadoras

1. Un programa es una lista de instrucciones de computadora organizadas para alcanzar un resultado específico.
2. Un código de operación es el código de una instrucción.
3. Una cadena de caracteres es una secuencia contigua de bytes o palabras.

SECCIÓN 12.5 Interrupciones

1. En la E/S controlada por interrupción, la CPU proporciona servicio a los periféricos sólo cuando el periférico lo solicita. En la E/S por sondeo, la CPU comprueba periódicamente el periférico para ver si necesita servicio.
2. Las E/S controladas por interrupción ahorran tiempo de CPU.
3. Una interrupción software es una instrucción que invoca una rutina de servicio de interrupción.

SECCIÓN 12.6 Acceso directo a memoria (DMA)

1. DMA (*Direct Memory Access*) quiere decir acceso directo a memoria.
2. Una transferencia de datos DMA desde la memoria o a una E/S o viceversa ahorra tiempo de CPU. A menudo, el acceso directo a memoria se utiliza para transferir datos entre la RAM y una unidad de disco.

SECCIÓN 12.7 Interfaces internas

1. Los excitadores triestado permiten a los dispositivos desconectarse por completo del bus cuando no están en uso, evitando la interferencia con otros dispositivos.
2. Un bus interconecta todos los dispositivos de un sistema y hace posible la comunicación entre los posibles dispositivos.

SECCIÓN 12.8 Buses estándar

1. Transferencia de datos serie y paralelo.
2. **(a)** paralelo **(b)** serie **(c)** serie **(d)** paralelo.
3. Serie: un bit al mismo tiempo. Paralelo: 8 o más bits al mismo tiempo.
4. 127 dispositivos USB
5. El bus FireWire es más rápido que el USB.

PROBLEMAS RELACIONADOS

12.1 $6C4C2_{16}$

12.2 Cambiar el primer bloque (bloque de inicialización) a "BIG = FFFF"; éste es el número sin signo más largo posible. Cambiar la primera pregunta por "¿Es el número < BIG?"

AUTOTEST

1. (c)	**2.** (b)	**3.** (d)	**4.** (b)	**5.** (c)	**6.** (b)	**7.** (c)	**8.** (b)
9. (a)	**10.** (d)	**11.** (b)	**12.** (c)	**13.** (c)	**14.** (a)	**15.** (b)	**16.** (b)
17. (b)	**18.** (b)	**19.** (c)	**20.** (d)				

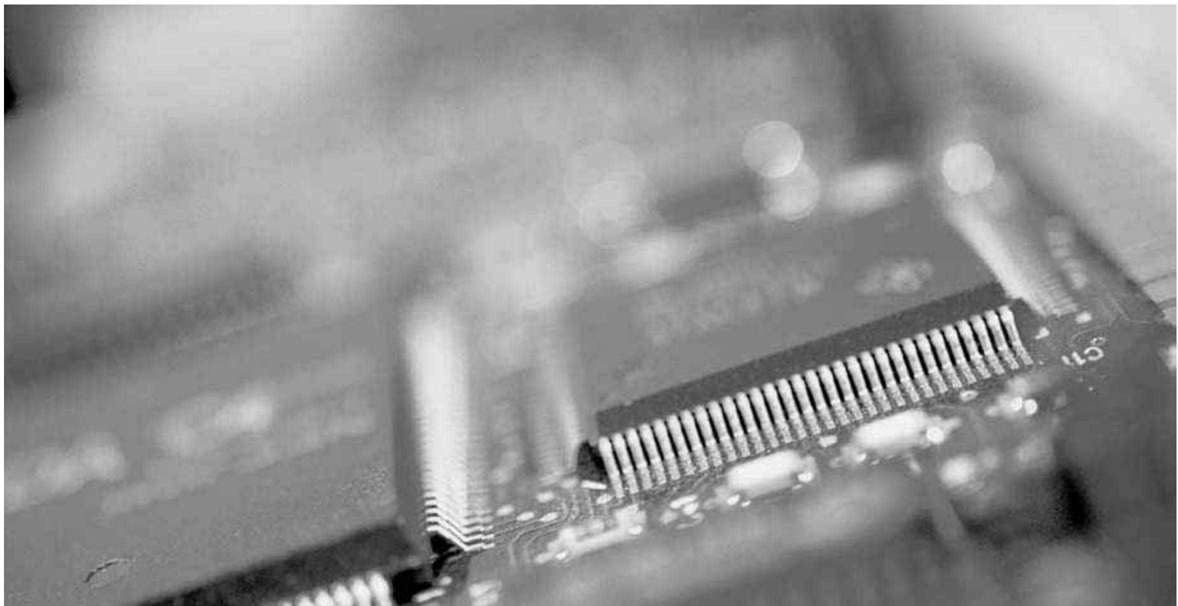
13

INTRODUCCIÓN AL PROCESAMIENTO DIGITAL DE LA SEÑAL

CONTENIDO DEL CAPÍTULO

OBJETIVOS DEL CAPÍTULO

- Enumerar los elementos esenciales en un sistema de procesamiento digital de la señal.
- Explicar cómo se convierten las señales analógicas en formato digital.
- Explicar el propósito del filtrado.
- Describir el proceso de muestreo.
- Establecer el propósito de la conversión analógica-digital.



- Explicar cómo operan diversos tipos de convertidores ADC.
- Explicar los conceptos básicos de un procesador digital de la señal (DSP)
- Describir la arquitectura básica de un DSP.
- Nombrar algunas de las funciones que realiza un DSP.
- Establecer el propósito de la conversión digital-analógica.
- Explicar cómo operan los convertidores DAC.

PALABRAS CLAVE

- Convertidor analógico-digital (ADC)
- DSP
- Convertidor digital-analógico (DAC)
- Muestreo
- Frecuencia de Nyquist
- *Aliasing*
- Cuantificación
- Módulo DSP
- MIPS
- MFLOPS
- MMACS
- *Pipeline*
- Extracción
- Decodificación
- Ejecución

INTRODUCCIÓN

El procesamiento digital de la señal es una potente tecnología ampliamente utilizada en muchas aplicaciones, como por ejemplo las de automoción, electrónica de consumo, tratamiento de gráficos/imágenes, electrónica industrial, instrumentación, medicina, tecnología militar, telecomunicaciones y aplicaciones de tratamiento de voz/habla. El procesamiento digital de la señal aglutina conceptos matemáticos, tecnologías de programación software y hardware de procesamiento para manipular señales analógicas. Por ejemplo, el procesamiento digital de la señal puede utilizarse para realzar imágenes, comprimir los datos para su eficiente transmisión y almacenamiento, reconocer y generar señales de voz y limpiar señales de audio ruidosas o deterioradas.

Este capítulo proporciona una breve panorámica del procesamiento digital de la señal. Para analizar de manera completa el tema con el detalle necesario como para comprender detalladamente todos estos temas haría falta mucho más que un simple capítulo. Sobre este tema hay disponibles libros completos y el lector podrá encontrar una lista al final del capítulo. En el sitio web de Texas Instruments (www.ti.com) podrá encontrar cuantiosa información, incluyendo las hojas de características sobre la familia TMS320 de procesadores DSP. Puede encontrar también información sobre otros DSP en el sitio web de Motorola (www.motorola.com) y en el sitio web de Analog Devices (www.analogdevices.com).

CIRCUITOS DE FUNCIÓN FIJA

ADC0804

PROCESADORES DIGITALES DE LA SEÑAL

TMS320C62xx TMS320C64xx TMS320C67xx

13.1 FUNDAMENTOS DEL PROCESAMIENTO DIGITAL DE LA SEÑAL

El procesamiento digital de la señal convierte señales de naturaleza analógica, tales como el sonido, el vídeo e información procedente de sensores, en formato digital, utilizando técnicas digitales para mejorar y modificar los datos de las señales analógicas para distintas aplicaciones.

Al finalizar esta sección el lector será capaz de:

■ Definir el concepto de *ADC*. ■ Definir el concepto de *DSP*. ■ Definir el concepto de *DAC*. ■ Dibujar un diagrama de bloques básico de un sistema de procesamiento digital de la señal.

Un sistema de procesamiento digital de la señal traduce primero una señal analógica que varía de manera continua a una serie de niveles discretos. Esta serie de niveles sigue las variaciones de la señal analógica y se asemeja a una escalera, como se ilustra para el caso de una onda sinusoidal en la Figura 13.1. El proceso de modificar la señal analógica original, obteniendo una aproximación "en escalera" de la misma, se realiza mediante un circuito de muestreo y retención.

A continuación, la aproximación "en escalera" se cuantifica para obtener una serie de códigos binarios que representan cada uno de los pasos discretos de esa aproximación, mediante un proceso denominado conversión analógico-digital (A/D). El circuito que realiza la conversión A/D se denomina ***convertidor analógico-digital*** (***ADC***, *Analog-to-Digital Converter*).

Una vez convertida la señal analógica a formato con codificación binaria, se la aplica a un ***procesador digital de la señal*** (***DSP***, *Digital Signal Proccesor*). El DSP puede realizar diversas operaciones con los datos entrantes, como por ejemplo eliminar las interferencias no deseadas, aumentar la amplitud de ciertas frecuencias de la señal y reducir la de otras, codificar los datos para realizar una transmisión segura de los mismos y detectar y corregir errores en los códigos transmitidos. Un DSP permite, entre otras muchas cosas, limpiar grabaciones sonoras, eliminar los ecos de las líneas de comunicaciones, realizar las imágenes de las tomografías computerizadas para mejorar el diagnóstico médico y cifrar las conversaciones de los teléfonos móviles para garantizar la intimidad.

Después de procesar una señal mediante DSP, la señal puede convertirse de nuevo a forma analógica, obteniéndose una señal muy mejorada de la señal analógica original. Este paso se lleva a cabo mediante un ***convertidor digital-analógico*** (***DAC***, *Digital-to-Analog Converter*). La Figura 13.2 muestra un diagrama de bloques básico de un sistema típico de procesamiento digital de la señal.

Un DSP es, de hecho, un tipo especializado de microprocesador, aunque difiere de los microprocesadores de propósito general en dos aspectos significativos. Típicamente, los microprocesadores están diseñados para

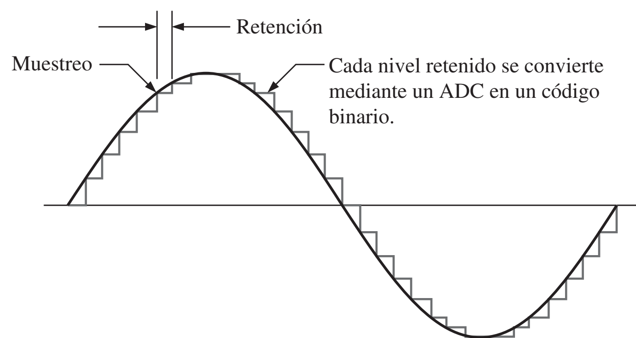


FIGURA 13.1 Una señal analógica original (onda sinusoidal) y su aproximación "en escalera".

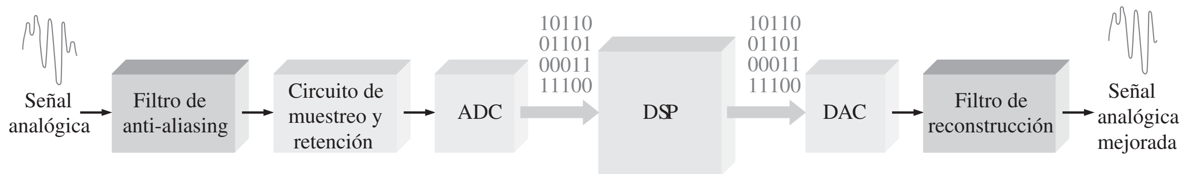


FIGURA 13.2 Diagrama de bloques básico de un sistema típico de procesamiento digital de la señal.

funciones de propósito general y operan con paquetes software de gran envergadura. Un DSP por el contrario se emplea para operaciones de propósito especial; se trata de un procesador numérico muy rápido que debe trabajar en tiempo real, procesando la información a medida que se genera, utilizando algoritmos (programas) especializados. El convertidor analógico-digital (ADC) de un sistema debe muestrear los datos analógicos entrantes a una frecuencia lo suficientemente alta como para capturar todas las fluctuaciones relevantes de la amplitud de la señal, y el DSP debe ajustarse a la frecuencia de muestreo del ADC, llevando a cabo sus cálculos con la misma rapidez con que se reciben los datos muestreados. Una vez que el DSP ha procesado los datos digitales, éstos pasan al convertidor digital-analógico (DAC) para volverlos a convertir a forma analógica.

REVISIÓN DE LA SECCIÓN 13.1

Las respuestas se encuentran al final del capítulo

1. ¿Qué significan las siglas DSP?
2. ¿Qué significan las siglas ADC?
3. ¿Qué significan las siglas DAC?
4. ¿Qué circuito transforma una señal analógica a un formato con codificación binaria?
5. ¿Qué circuito transforma una señal con codificación binaria a forma analógica?

13.2 CONVERSIÓN DE SEÑALES ANALÓGICAS A FORMATO DIGITAL

Para poder procesar las señales utilizando técnicas digitales, la señal analógica entrante debe convertirse a forma digital.

Al finalizar esta sección el lector será capaz de:

■ Explicar el proceso básico de conversión de una señal analógica en digital. ■ Describir el propósito de la función de muestreo y retención. ■ Definir la frecuencia de Nyquist. ■ Indicar por qué aparece el fenómeno del *aliasing* y explicar cómo se puede eliminar. ■ Describir el propósito de un ADC.

Muestreo y filtrado

Los dos primeros bloques del diagrama de sistemas de la Figura 13.2 son el filtro de *anti-aliasing* y el circuito de muestreo y retención. La función de muestreo y retención lleva a cabo dos operaciones, la primera de las cuales es el muestreo. El ***muestreo*** es el proceso de tomar un número suficiente de valores discretos en determinados puntos de una forma de onda como para poder definir adecuadamente esa forma de onda. Cuantas más muestras se tomen, más precisamente se podrá definir esa forma de onda. El muestreo convierte una señal analógica en una serie de impulsos, cada uno de los cuales representa la amplitud de la señal en un determinado instante. La Figura 13.3 ilustra el proceso de muestreo.

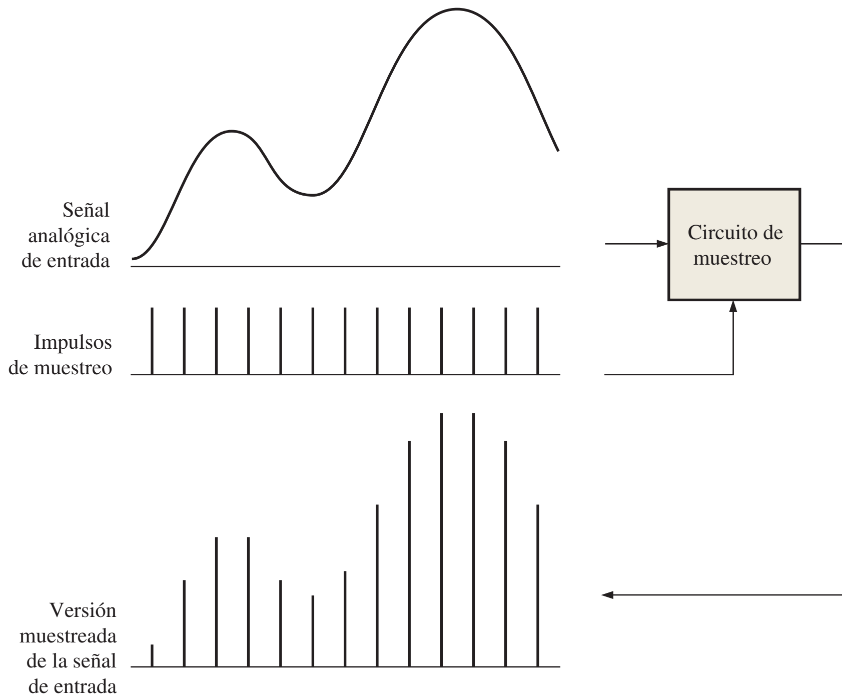


FIGURA 13.3 Ilustración del proceso de muestreo.

Cuando hay que muestrear una señal analógica, es necesario satisfacer ciertos criterios para poder representar de forma precisa la señal original. Todas las señales analógicas (excepto las ondas sinusoidales puras) contienen un espectro de frecuencias componentes, denominadas *armónicos*. Los armónicos de una señal analógica son ondas sinusoidales de diferentes frecuencias y amplitudes. Al sumar todos los armónicos de una cierta forma de onda periódica, el resultado es la señal original. Antes de poder muestrear una señal, es preciso pasarla a través de un filtro paso-bajo (filtro de *anti-aliasing*) para eliminar las frecuencias armónicas situadas por encima de un cierto valor, que estará determinado por la frecuencia de Nyquist.

El teorema de muestreo Observe en la Figura 13.3 que hay dos formas de onda de entrada: la señal analógica y la forma de onda que representa los pulsos de muestreo. El teorema de muestreo establece que para poder representar una señal analógica, la frecuencia de muestreo, f_{muestreo}, debe ser al menos dos veces superior a la componente de mayor frecuencia $f_{\text{a(máx)}}$ de la señal analógica. Otra forma de indicar este hecho es que la frecuencia analógica más alta no puede ser mayor que la mitad de la frecuencia de muestreo. La frecuencia $f_{\text{a(máx)}}$ se conoce como la ***frecuencia de Nyquist*** y está expresada en la Ecuación 13.1. En la práctica, la frecuencia de muestreo debe ser dos veces superior a la frecuencia analógica más alta.

Ecuación 13.1 $$f_{\text{muestreo}} \geq 2f_{\text{a(máx)}}$$

Para comprender de manera intuitiva el teorema de muestreo, nos puede resultar de utilidad una analogía simple basada en el rebote de una pelota. Aunque no se trata de una representación perfecta del proceso de muestreo de una señal eléctrica, sí que nos servirá para ilustrar el concepto básico. Si fotografíamos (muestreamos) una pelota en un determinado instante mientras efectúa un único rebote, como se ilustra en la Figura 13.4(a), no podemos extraer ninguna conclusión acerca de la trayectoria seguida por la pelota; lo único que podemos decir es que ésta se encuentra por encima del suelo. No podemos saber si está yendo hacia arriba o hacia abajo ni tampoco podemos determinar la altura hasta la que ha rebotado. Si tomamos sendas fotogra-

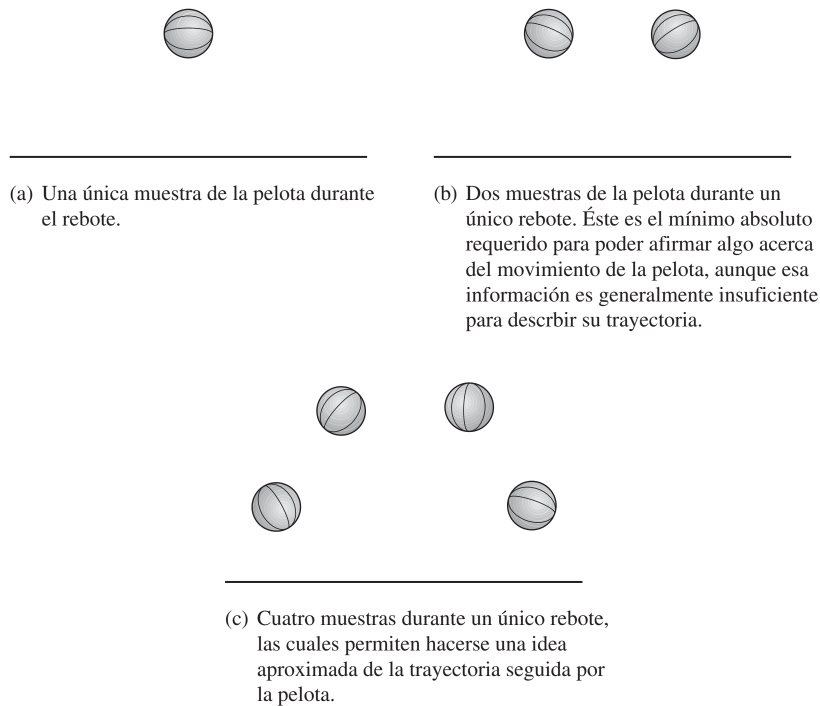


FIGURA 13.4 Analogía de la teoría de muestreo.

fías en dos instantes equiespaciados durante un mismo rebote, como se muestra en la parte (b) de la figura, podemos tan sólo obtener una cantidad mínima de información acerca del movimiento de la pelota, y no podemos obtener información ninguna acerca de la distancia hasta la que ha rebotado. En este caso concreto, sólo sabemos que la pelota estaba en el aire en los instantes en que se tomaron esas dos fotografías, y que la altura máxima de rebote es al menos igual a la altitud que en esas fotografías se muestra. Si tomamos cuatro fotografías, como se muestra en la parte (c) de la figura, comienza a resultar aparente la trayectoria seguida por la pelota durante un rebote. Cuantas más fotografías (muestras) tomemos más precisamente podremos determinar la trayectoria de la pelota durante su rebote en el suelo.

La necesidad del filtrado. El filtrado paso-bajo es necesario para eliminar todas las componentes de frecuencia (armónicos) de la señal analógica que excedan la frecuencia de Nyquist. Si hay alguna componente de frecuencia en la señal analógica que sea superior a la frecuencia de Nyquist, se produce una condición indeseada que se conoce con el nombre de ***aliasing***. Un alias es una señal que aparece cuando la frecuencia de muestreo no es al menos igual a dos veces la frecuencia máxima de la señal. La señal de alias tiene una frecuencia que es inferior a la frecuencia más alta de la señal analógica que se está muestreando y que, por tanto, cae dentro de la banda de frecuencias de la señal de entrada analógica, provocando la distorsión de ésta. Esta señal espúrea aparece en la práctica como parte de la señal analógica cuando realmente no lo es; de ahí que se utilice el término *alias*.

Otra forma de considerar el fenómeno del *aliasing* consiste en tener en cuenta que los pulsos de muestreo producen un espectro de frecuencias armónicas situado por encima y por debajo de la frecuencia de muestreo, como se ilustra en la Figura 13.5. Si la señal analógica contiene frecuencias por encima de la frecuencia de Nyquist, dichas frecuencias se solaparán en el espectro de la forma de onda muestreada, tal como revela la figura, produciéndose una interferencia. Las componentes de baja frecuencia de la forma de onda de mues-

treo se mezclan con el espectro de frecuencia de la forma de onda analógica, dando como resultado un error de *aliasing*.

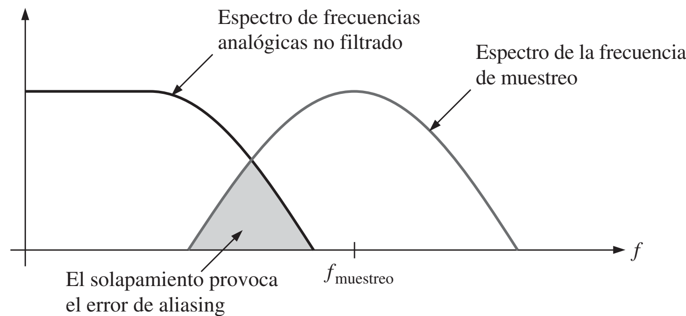


FIGURA 13.5 Ilustración básica de la condición $f_{muestreo} < 2f_{a(máx)}$.

Es necesario utilizar un filtro *antialiasing* paso-bajo para limitar el espectro de frecuencias de la señal analógica, dada una cierta frecuencia de muestreo. Para evitar el error de *aliasing*, el filtro debe eliminar al menos todas las frecuencias analógicas situadas por encima de la frecuencia mínima de muestreo, como se ilustra en la Figura 13.6. El *aliasing* puede también evitarse incrementando de forma suficiente la frecuencia de muestreo. Sin embargo, la frecuencia máxima de muestreo está limitada usualmente por las prestaciones del convertidor analógico-digital (ADC) situado a continuación.

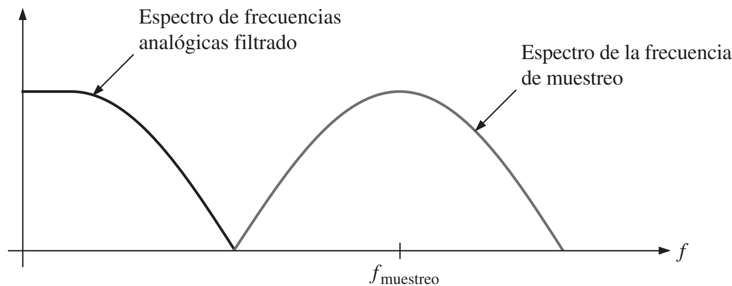


FIGURA 13.6 Después del filtrado paso-bajo, el espectro de frecuencias de la señal analógica y de la señal de muestreo no se solapan, eliminando así el error de *aliasing*.

Aplicación Un ejemplo de aplicación de las técnicas de muestreo es el de los equipos digitales de sonido. Las frecuencias de muestreo utilizadas son 32 kHz, 44,1 kHz o 48 kHz (el número de muestras por segundo). La frecuencia más común es la de 48 kHz, pero la frecuencia de 44,1 kHz se utiliza para los CD de audio y las cintas magnéticas pregrabadas. De acuerdo con la frecuencia de Nyquist, la frecuencia de muestreo tiene que ser igual a dos veces la frecuencia máxima de la señal de audio. Por tanto, la frecuencia de muestreo típica de un CD de 44,1 kHz, permite capturar frecuencias de hasta unos 22 kHz, lo que supera los 20 kHz que comúnmente se especifican para la mayoría de los equipos de sonido. Muchas aplicaciones no requieren un amplio rango de frecuencias para poder reproducir un sonido aceptable. Por ejemplo, la voz humana contiene algunas frecuencias próximas a los 10 kHz y requiere, por tanto, una frecuencia de muestreo de al menos 20 kHz. Sin embargo, si sólo reproducimos las frecuencias hasta 4 kHz (lo que idealmente requeriría una frecuencia mínima de muestreo de 8 kHz), la voz sigue siendo perfectamente comprensible. Por el contrario, si no se

muestrea una señal sonora a una frecuencia lo suficientemente alta, el efecto de *aliasing* comenzará a ser perceptible apareciendo ruido de fondo y distorsión.

Retención del valor muestreado

La operación de retención es parte del bloque de muestreo y retención mostrado en la Figura 13.2. Después del filtrado y del muestreo, el nivel muestreado debe mantenerse constante hasta que se tome la siguiente muestra. Esto es necesario para que el ADC disponga del suficiente tiempo como para procesar el valor muestreado. Esta operación de muestreo y retención genera una forma de onda "en escalera" que se aproxima a la forma de onda analógica de entrada, como se muestra en la Figura 13.7.

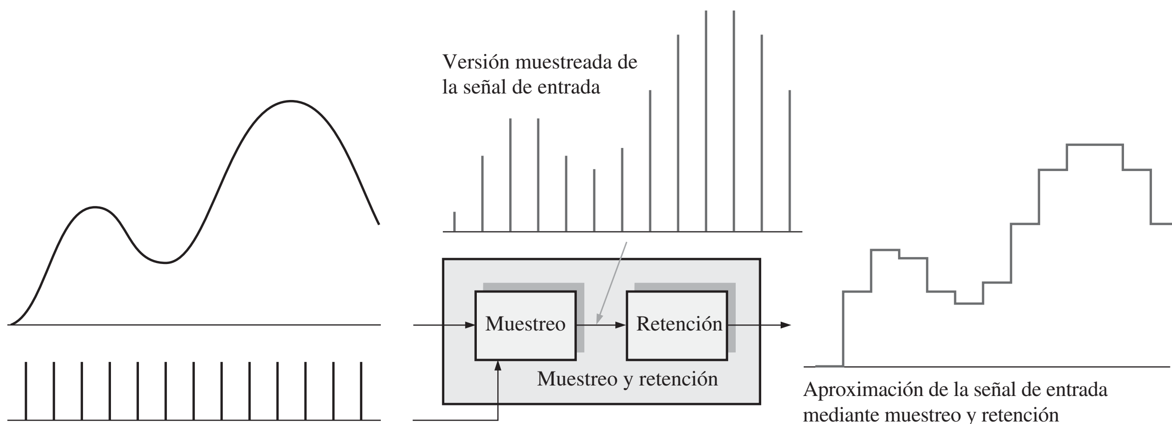


FIGURA 13.7 Ilustración de la operación de muestreo y retención.

Conversión analógico-digital

La conversión analógico-digital es el proceso de convertir la salida del circuito de muestreo y retención en una serie de códigos binarios que representan la amplitud de la entrada analógica en cada uno de los instantes de muestreo. El proceso de muestreo y retención hace que se mantenga constante la amplitud de la señal analógica de entrada entre sucesivos impulsos de muestreo; así, la conversión analógico-digital puede realizarse utilizando un valor constante, en lugar de permitir que la señal analógica varíe durante el intervalo de conversión, que es el intervalo comprendido entre los impulsos de muestreo. La Figura 13.8 ilustra la función básica de un convertidor analógico-digital (ADC). Los intervalos de muestreo se indican mediante líneas de puntos.

Cuantificación El proceso de convertir un valor analógico en un determinado código se denomina ***cuantificación***. Durante el proceso de cuantificación, el ADC convierte cada valor muestreado de la señal analógica en un código binario. Cuantos más bits se empleen para representar un valor muestreado, más precisa será la representación.

Como ilustración, vamos a cuantizar una reproducción de la forma de onda analógica en cuatro niveles (0-3). Como se muestra en la figura 13.9, hacen faltan dos bits. Observe que cada nivel de cuantificación se representa mediante un código de dos bits en el eje vertical, mientras que cada intervalo de muestreo está numerado a lo largo del eje horizontal. El proceso de cuantificación se resume en la Tabla 13.1.

Si utilizamos los códigos digitales resultantes de 2 bits para reconstruir la forma de onda original, lo que se puede realizar mediante un convertidor digital-analógico (DAC), obtendremos la forma de onda mostrada

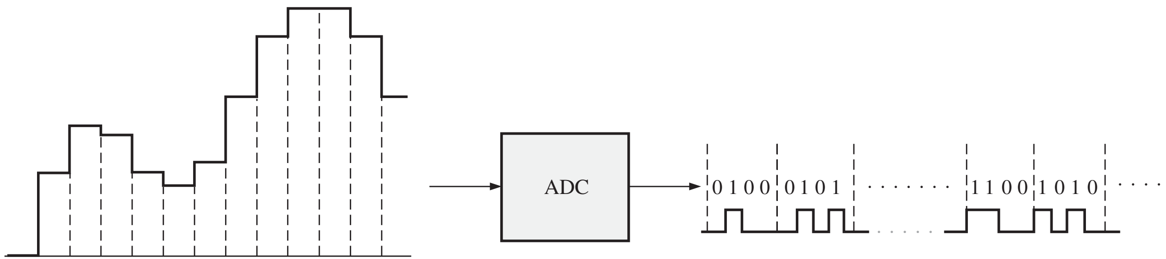


FIGURA 13.8 Función básica de un convertidor analógico-digital (ADC). Los códigos binarios y los números de bits se han elegido arbitrariamente con propósitos de ilustración. La figura también muestra la forma de onda de salida del ADC, que representa los códigos binarios.

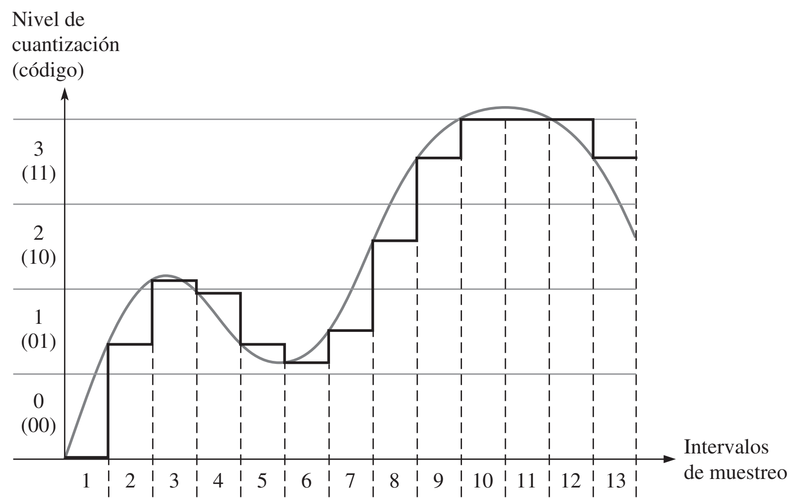


FIGURA 13.9 Forma de onda de salida del bloque de muestreo y retención, con cuatro niveles de cuantificación. Se muestra como referencia la forma de onda analógica original.

Intervalo de muestreo	Nivel de cuantificación	Código	Intervalo de muestreo	Nivel de cuantificación	Código
1	0	00	8	2	10
2	1	01	9	3	11
3	2	10	10	3	11
4	1	01	11	3	11
5	1	01	12	3	11
6	1	01	13	3	11
7	1	01			

TABLA 13.1 Cuantificación de dos bits para la forma de onda de la Figura 13.9.

en la Figura 13.10. Como puede verse, se pierde bastante precisión cuando sólo se emplean dos bits para representar los valores muestreados.

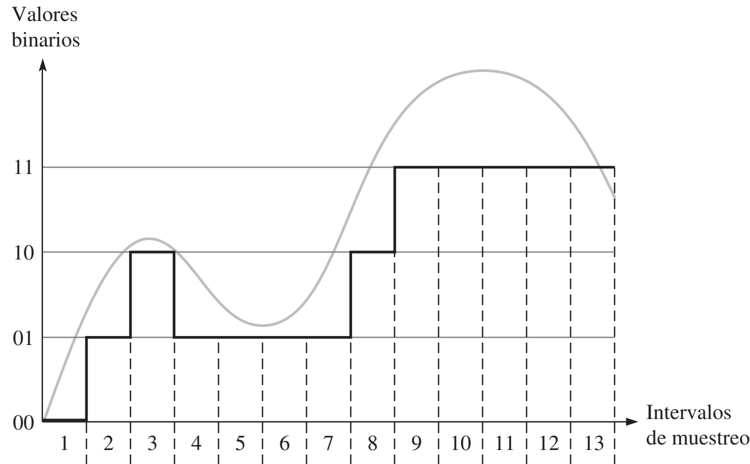


FIGURA 13.10 Forma de onda reconstruida de la Figura 13.9, utilizando cuatro niveles de cuantificación (2 bits). La forma de onda analógica original se muestra como referencia.

Ahora, vemos cómo mejora la precisión añadiendo más bits. La Figura 13.11 muestra la misma forma de onda con dieciséis niveles de cuantificación (4 bits). El proceso de cuantificación de 4 bits se resume en la Tabla 13.2.

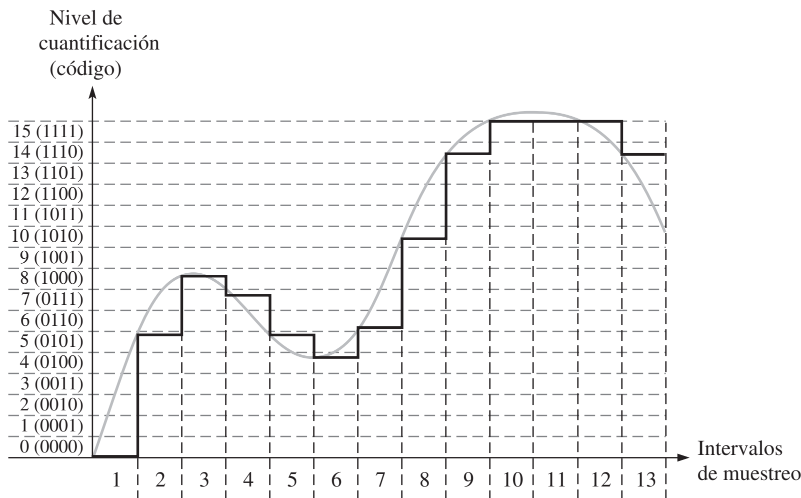


FIGURA 13.11 Forma de onda de salida del bloque de muestreo y retención con dieciséis niveles de cuantificación. También se muestra la forma de onda analógica original como referencia.

Si utilizamos los códigos digitales resultantes de 4 bits para reconstruir la forma de onda original, obtendremos la forma de onda mostrada en la Figura 13.12. Como puede verse, el resultado se parece bastante más a la forma de onda original que en el caso de los cuatro niveles de cuantificación representado en la Figura 13.10. Esto demuestra que puede conseguirse una mayor precisión empleando más bits de cuantificación. La mayoría de los ADC en circuito integrado utilizan entre 8 y 14 bits, y la función de muestreo y retención suele

estar incluida en el propio chip ADC. En la siguiente sección presentaremos varios tipos de convertidores ADC.

Intervalo de muestreo	Nivel de cuantificación	Código	Intervalo de muestreo	Nivel de cuantificación	Código
1	0	0000	8	10	1010
2	5	0101	9	14	1110
3	8	1000	10	15	1111
4	7	0111	11	15	1111
5	5	0101	12	15	1111
6	4	0100	13	14	1110
7	6	0110			

TABLA 13.2 Cuantificación de 4 bits para la forma de onda de la Figura 13.11.

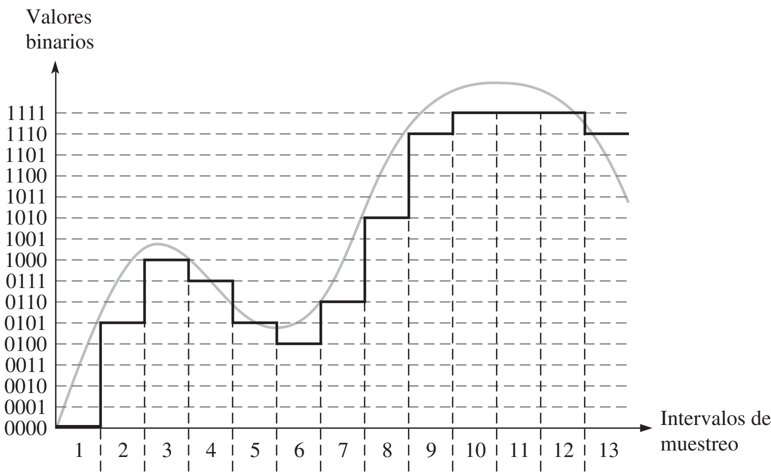


FIGURA 13.12 La forma de onda reconstruida de la Figura 13.11, utilizando dieciséis niveles de cuantificación (4 bits). También se muestra la forma de onda analógica original como referencia.

REVISIÓN DE LA SECCIÓN 13.2

1. ¿Qué quiere decir muestreo?
2. ¿Por qué es necesario retener cada valor muestreado?
3. Si la componente más alta de frecuencia en una señal analógica es de 20 kHz, ¿cuál es la frecuencia de muestreo mínima?
4. ¿Qué quiere decir cuantificación?
5. ¿Qué es lo que determina la precisión del proceso de cuantificación?

13.3 MÉTODOS DE CONVERSIÓN ANALÓGICA-DIGITAL

Como ya hemos visto, la conversión analógica-digital es el proceso por el que una magnitud analógica se convierte a formato digital. Este proceso es necesario cuando las magnitudes medidas deben estar

en formato digital para poder procesarlas, mostrarlas o almacenarlas. En esta sección vamos a examinar algunos tipos comunes de convertidores analógicos-digitales (ADC). Dos parámetros de gran importancia de los ADC son la *resolución*, que es el número de bits y la *tasa de transferencia*, que es la frecuencia de muestreo que un ADC puede aceptar, en número de muestras por segundo.

Al finalizar esta sección el lector será capaz de:

■ Explicar de manera básica qué es un amplificador operacional. ■ Mostrar cómo puede utilizarse un amplificador operacional como amplificador inversor o comparador. ■ Explicar cómo funciona un ADC flash. ■ Explicar el funcionamiento de un ADC de doble pendiente. ■ Describir la operación de un ADC de aproximaciones sucesivas. ■ Describir un ADC delta-sigma. ■ Explicar cómo se prueba un ADC para ver si faltan códigos, si hay códigos incorrectos o errores de *offset*.

El amplificador operacional

Antes de abordar los convertidores analógico-digitales (ADC), vamos a ver brevemente un elemento muy común en la mayoría de los ADC y de los convertidores digital-analógicos (DAC). Este elemento es el amplificador operacional (AO). A continuación tiene un breve resumen sobre este dispositivo.

Un **amplificador operacional** es un amplificador lineal que tiene dos entradas (inversora y no inversora) y una salida. Tiene una alta ganancia en tensión y una muy alta impedancia de entrada, así como una muy baja impedancia de salida. En la Figura 13.13(a) se muestra el símbolo del AO. Cuando se utiliza como amplificador inversor, se configura como se indica en la parte (b) de la figura. La resistencia de realimentación, R_f, y la resistencia de entrada, R_i, controlan la ganancia en tensión de acuerdo con la fórmula de la Ecuación 13.2, donde V_{out}/V_{in} es la ganancia en tensión en bucle cerrado (bucle cerrado se refiere a la realimentación desde la salida hasta la entrada por medio de R_f). El signo negativo indica inversión.

Ecuación 13.2

$$\frac{V_{out}}{V_{in}} = -\frac{R_f}{R_i}$$

En la configuración de amplificador inversor, la entrada inversora del AO está, aproximadamente, al potencial de tierra (0 V), porque la realimentación y la extremadamente alta ganancia en bucle abierto hacen que la diferencia de tensión entre las dos entradas sea muy pequeña. Por tanto, como la entrada no inversora está a tierra, la entrada inversora está aproximadamente a 0 V, lo que se denomina *tierra virtual.*

Cuando el AO se utiliza como comparador, como se muestra en la Figura 13.13(c) se aplican dos tensiones a las entradas. Cuando estas tensiones de entrada difieren en una pequeña cantidad, el AO pasa a uno de sus dos estados de salida saturados, nivel ALTO o BAJO, dependiendo de qué tensión sea mayor.

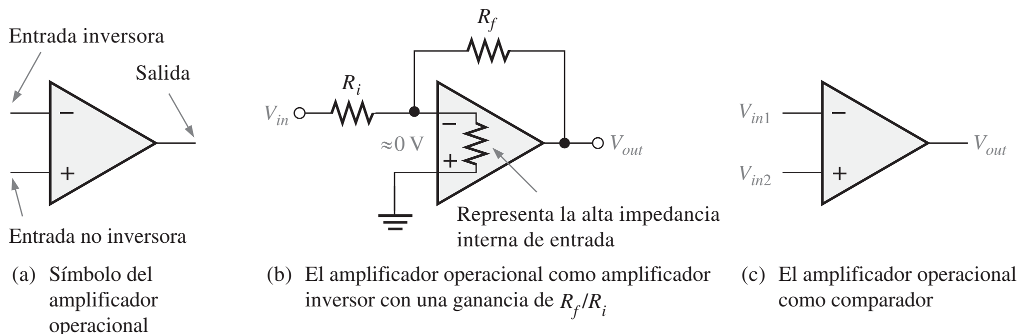


FIGURA 13.13 El amplificador operacional.

Convertidor analógico-digital flash (paralelo)

El método flash utiliza comparadores que comparan una serie de tensiones de referencia con la tensión de entrada analógica. Cuando la tensión analógica sobrepasa a la tensión de referencia de un comparador determinado, se genera un nivel ALTO. La Figura 13.14 presenta un convertidor de 3 bits que utiliza siete circuitos comparadores; no se necesita comparador para el caso de que todas las comparaciones sean cero. En general, se requieren 2^n-1 comparadores para la conversión a un código binario de n bits. El número de bits empleado en un ADC es su **resolución**. Una de las desventajas del **ADC flash** es el gran número de comparadores necesarios para un número binario de tamaño razonable. Su principal ventaja es que tiene un tiempo de conversión rápido, gracias a su alta tasa de transferencia, la cual se mide en muestras por segundo.

La tensión de referencia de cada comparador se establece mediante un circuito divisor de tensión resistivo. La salida de cada comparador se conecta a una entrada del codificador de prioridad. El codificador se habilita mediante un impulso aplicado a la entrada de habilitación *EN*, y el código de tres bits que representa el valor de la entrada analógica se presenta en las salidas del codificador. El código binario queda determinado por la entrada de mayor orden que se encuentre a nivel ALTO.

La frecuencia de los impulsos de habilitación y el número de bits del código binario determinan la precisión con la que la secuencia de códigos digitales representa la entrada del ADC. Debe haber un pulso de habilitación por cada nivel de muestreo de la señal de entrada.

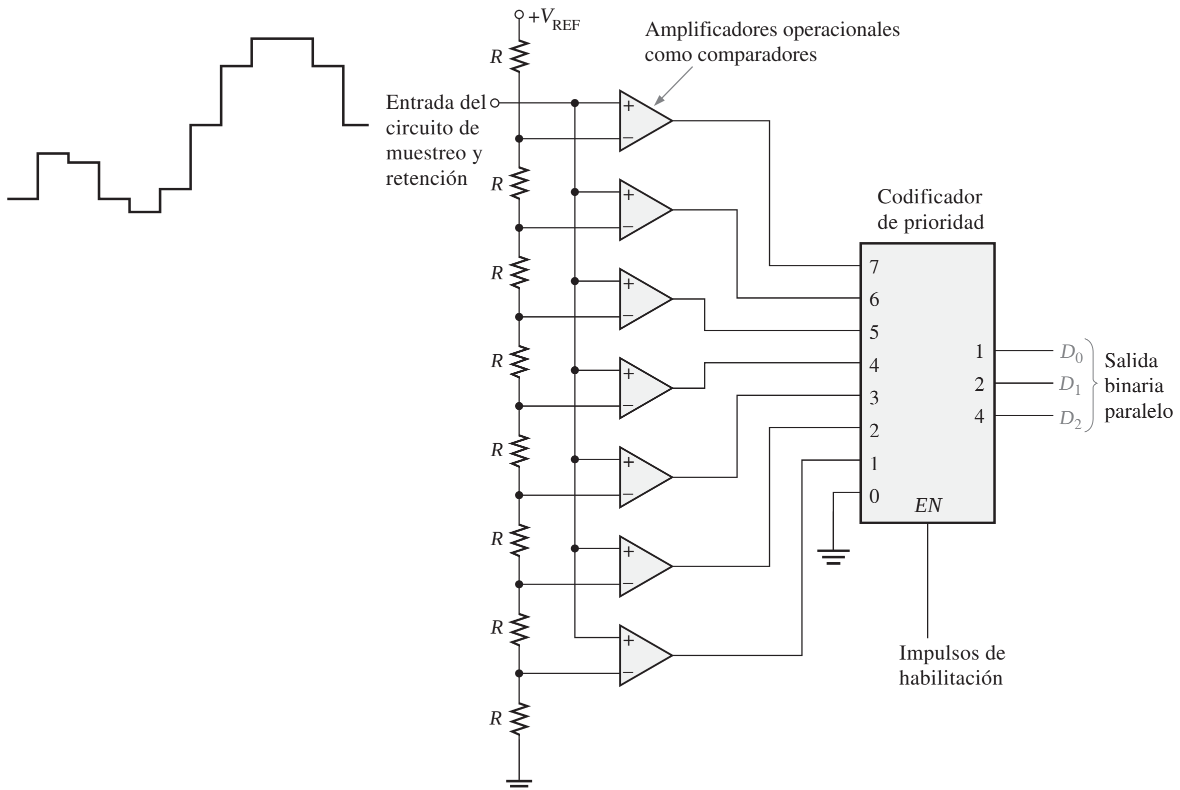


FIGURA 13.14 ADC flash de 3 bits.

EJEMPLO 13.1

Determinar el código binario de salida del ADC flash de tres bits de la Figura 13.14 para la señal analógica de entrada de la Figura 13.15 y los impulsos de habilitación del codificador mostrados. En este ejemplo, V_{REF} = +8 V.

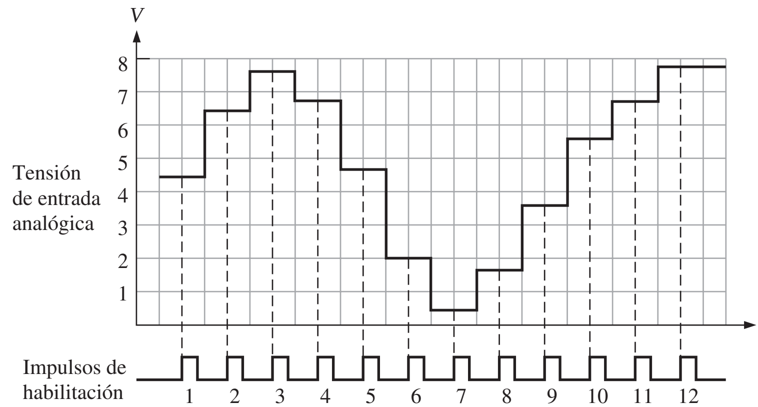


FIGURA 13.15 Muestreo de una forma de onda para convertirla a código binario.

Solución La secuencia digital de salida resultante corresponde a los valores siguientes y al diagrama de formas de onda que se muestra en la Figura 13.16 en relación con los impulsos de habilitación.

100, 110, 111, 110, 100, 010, 000, 001, 011, 101, 110, 111

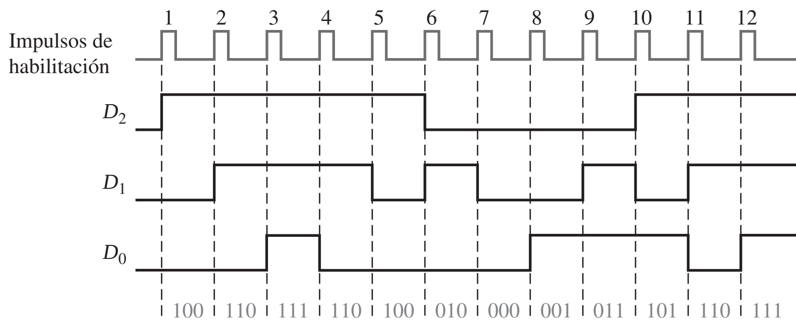


FIGURA 13.16 Salidas digitales resultantes para los valores del bloque de muestreo y retención. La salida D_0 es el LSB del código binario de 3 bits.

Problema relacionado* Si se duplica la frecuencia de los impulsos de habilitación de la Figura 13.15, determinar los números binarios representados por la secuencia digital de salida resultante, para 6 impulsos. ¿Se pierde alguna información?

* Las respuestas se encuentran al final del capítulo.

Convertidor analógico-digital de pendiente doble

El ADC de pendiente doble se utiliza comúnmente en voltímetros digitales y otros tipos de instrumentos de medida. Utilizan un generador de rampa (integrador) para generar la característica de pendiente doble. En la Figura 13.17 se presenta un diagrama de bloques de un ADC de pendiente doble.

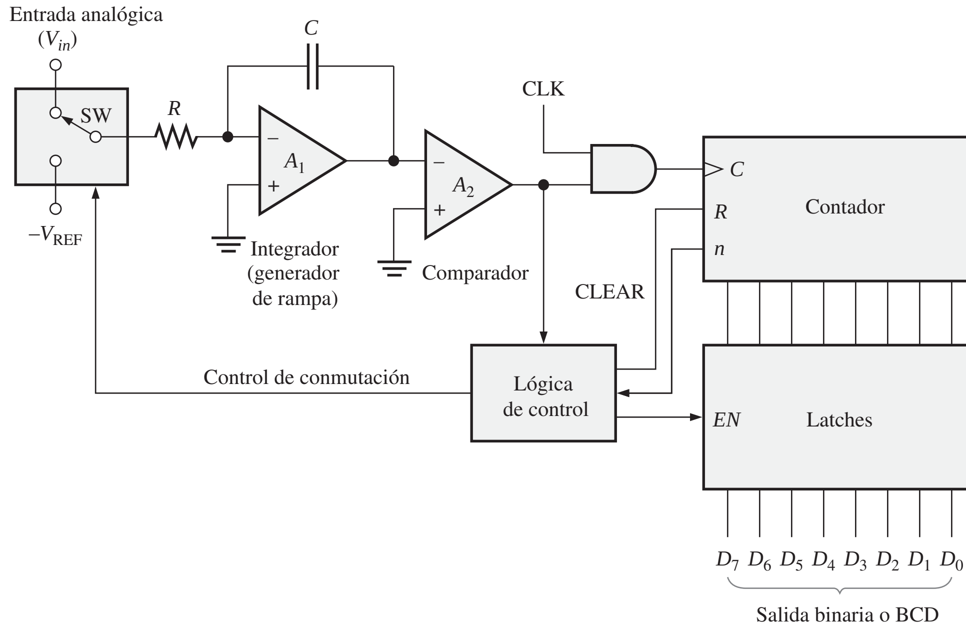
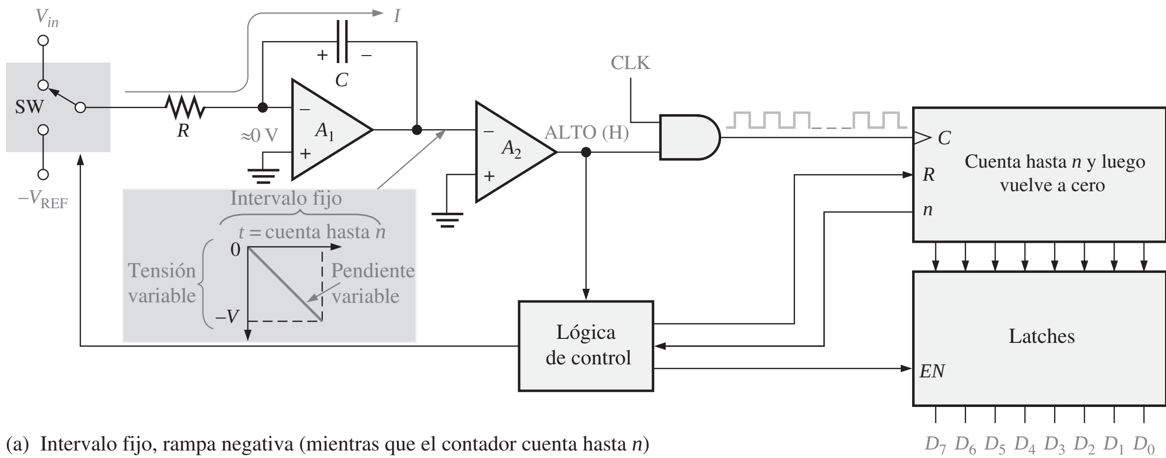


FIGURA 13.17 ADC de pendiente doble básico.

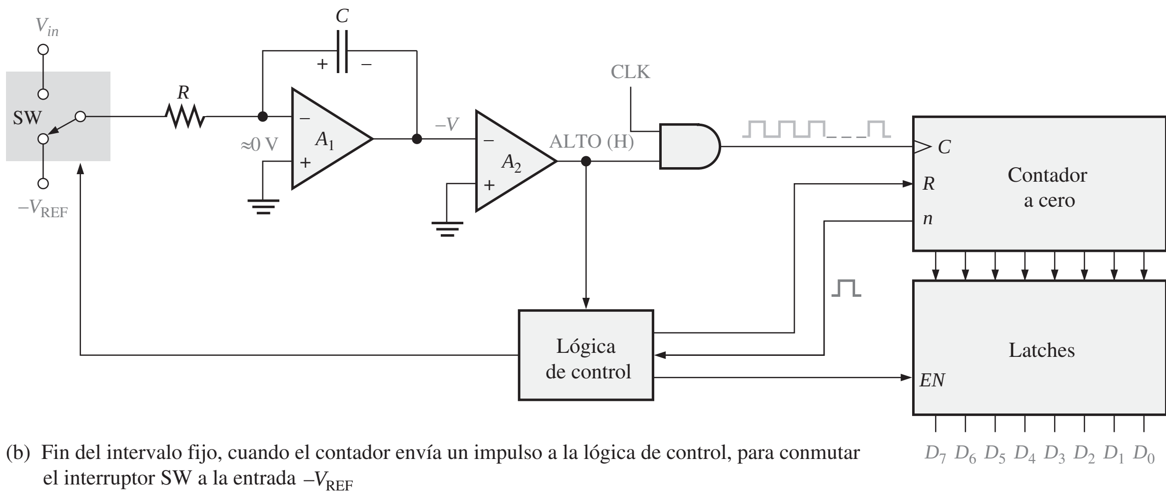
La Figura 13.18 ilustra la conversión de pendiente doble. Comencemos suponiendo que el contador está en estado RESET y la salida del integrador es cero. Ahora supongamos que se aplica a la entrada una tensión de entrada positiva por medio del interruptor (SW), comandado por la lógica de control. Puesto que la entrada inversora de A_1 está a tierra virtual y suponiendo que V_{in} es constante durante un período de tiempo, circulará una corriente constante a través de la resistencia de entrada R y, por tanto, a través del condensador C. El condensador C se cargará linealmente ya que la corriente es constante y, como resultado, habrá una rampa lineal negativa de tensión en la salida de A_1, como se muestra en la Figura 13.18(a).

Cuando el contador alcance un valor de cuenta especificado, se pondrá a cero (RESET) y la lógica de control conmutará a la tensión de referencia negativa ($-V_{REF}$) aplicándola a la entrada de A_1, como muestra la Figura 13.18(b). En este instante, el condensador está cargado a una tensión negativa ($-V$) proporcional a la tensión analógica de entrada.

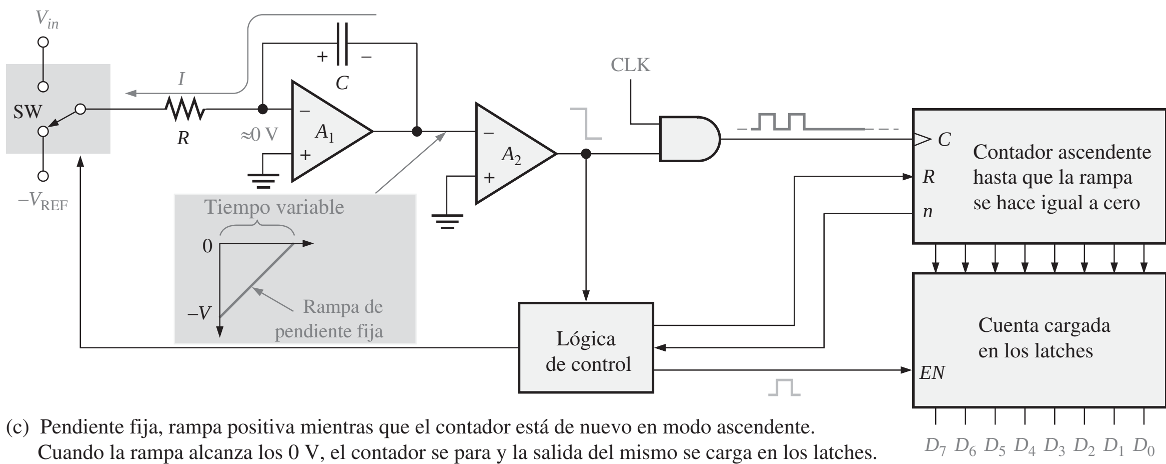
A continuación, el condensador se descarga linealmente debido a la corriente constante de $-V_{REF}$, como ilustra la Figura 13.18(c). Esta descarga lineal produce una rampa positiva en la salida de A_1, cuyo valor inicial es $-V$ y que tiene una pendiente constante, independiente de la tensión de carga. A medida que el condensador se descarga, el contador avanza desde su estado de RESET. El tiempo que tarda el condensador en descargarse hasta cero depende de la tensión inicial $-V$ (proporcional a V_{in}), puesto que la velocidad (pendiente) de descarga es constante. Cuando la tensión de salida del integrador (A_1) alcanza el valor cero, el comparador A_2 conmuta al estado BAJO e inhabilita la señal de reloj aplicada al contador. La cuenta binaria se almacena en los *latches*, completando un ciclo de conversión. La cuenta binaria es proporcional a V_{in}, ya que el tiempo que tarda el condensador en descargarse sólo depende de $-V$, y el contador registra este intervalo de tiempo.



(a) Intervalo fijo, rampa negativa (mientras que el contador cuenta hasta *n*)



(b) Fin del intervalo fijo, cuando el contador envía un impulso a la lógica de control, para conmutar el interruptor SW a la entrada $-V_{REF}$



(c) Pendiente fija, rampa positiva mientras que el contador está de nuevo en modo ascendente. Cuando la rampa alcanza los 0 V, el contador se para y la salida del mismo se carga en los latches.

FIGURA 13.18 Ilustración de la conversión de pendiente doble.

Convertidor analógico-digital por aproximaciones sucesivas

Quizás el método de conversión A/D más ampliamente utilizado es el de las aproximaciones sucesivas. Tiene un tiempo de conversión mucho menor que la conversión de pendiente doble, aunque es más lento que el método flash. Asimismo, el tiempo de conversión es fijo para cualquier valor de la entrada analógica.

La Figura 13.19 muestra un diagrama de bloques básico de un ADC por aproximaciones sucesivas de 4 bits. Está formado por un DAC (los DAC se estudian en la Sección 13.5), un registro de aproximaciones sucesivas (SAR, *Successive-Approximation Register*) y un comparador. Su funcionamiento básico es el siguiente: los bits de entrada al DAC se habilitan (se ponen a 1) de uno en uno sucesivamente, comenzando por el bit más significativo (MSB). Cada vez que se habilita un bit, el comparador produce una salida que indica si la tensión analógica de entrada es mayor o menor que la salida del DAC. Si la salida del DAC es mayor que la señal de entrada, la salida del comparador está a nivel BAJO, haciendo que el bit en el registro pase a cero. Si la salida es menor que la entrada, el bit 1 se mantiene en el registro. El sistema realiza esta operación con el MSB primero, luego con el siguiente bit más significativo, después con el siguiente, y así sucesivamente. Después de que todos los bits del DAC hayan sido aplicados, el ciclo de conversión estará completo.

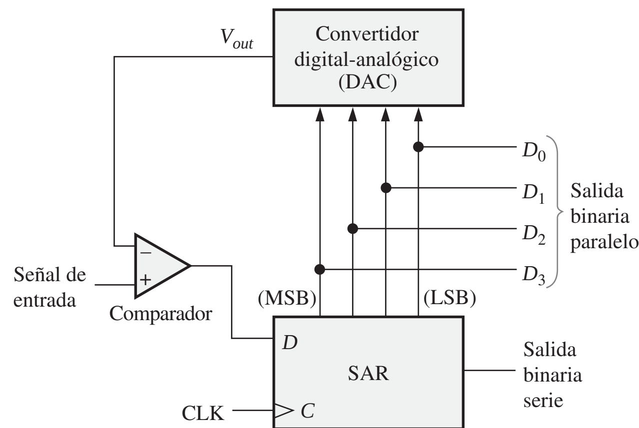


FIGURA 13.19 ADC por aproximaciones sucesivas.

Para comprender mejor el funcionamiento del ADC por aproximaciones sucesivas, vamos a realizar un ejemplo específico de una conversión de 4 bits. La Figura 13.20 ilustra la conversión paso a paso de una tensión de entrada constante (en este caso, 5,1 V). Suponemos que el DAC tiene las siguientes características de salida: V_{out} = 8 V para el bit 2^3 (MSB), V_{out} = 4 V para el bit 2^2, V_{out} = 2 V para el bit 2^1 y V_{out} = 1 V para el bit 2^0 (LSB).

La Figura 13.20(a) muestra el primer paso del ciclo de conversión con el MSB = 1. La salida del DAC es 8 V. Puesto que es mayor que la entrada de 5,1 V, la salida del comparador está a nivel BAJO, lo que hace que el MSB del SAR se ponga a 0.

La Figura 13.20(b) muestra el segundo paso del ciclo de conversión, con el bit 2^2 igual a 1. La salida del DAC es 4 V. Puesto que es menor que la entrada de 5,1 V, la salida del comparador conmuta a nivel ALTO, lo que hace que este bit se mantenga en el SAR.

La Figura 13.20(c) muestra el tercer paso del ciclo de conversión, con el bit 2^1 igual a 1. La salida del DAC es 6 V, ya que los bits de entrada 2^2 y 2^1 están a 1; (4V + 2 V = 6 V). Puesto que es mayor que la entrada de 5,1 V, la salida del comparador conmuta a nivel BAJO, lo que hace que este bit se ponga a 0.

La Figura 13.20(d) muestra el cuarto y último paso del ciclo de conversión, con el bit 2^0 igual a 1. La salida del DAC es 5 V, ya que los bits de entrada 2^1 y 2^0 están a 1; (4V + 1 V = 5 V).

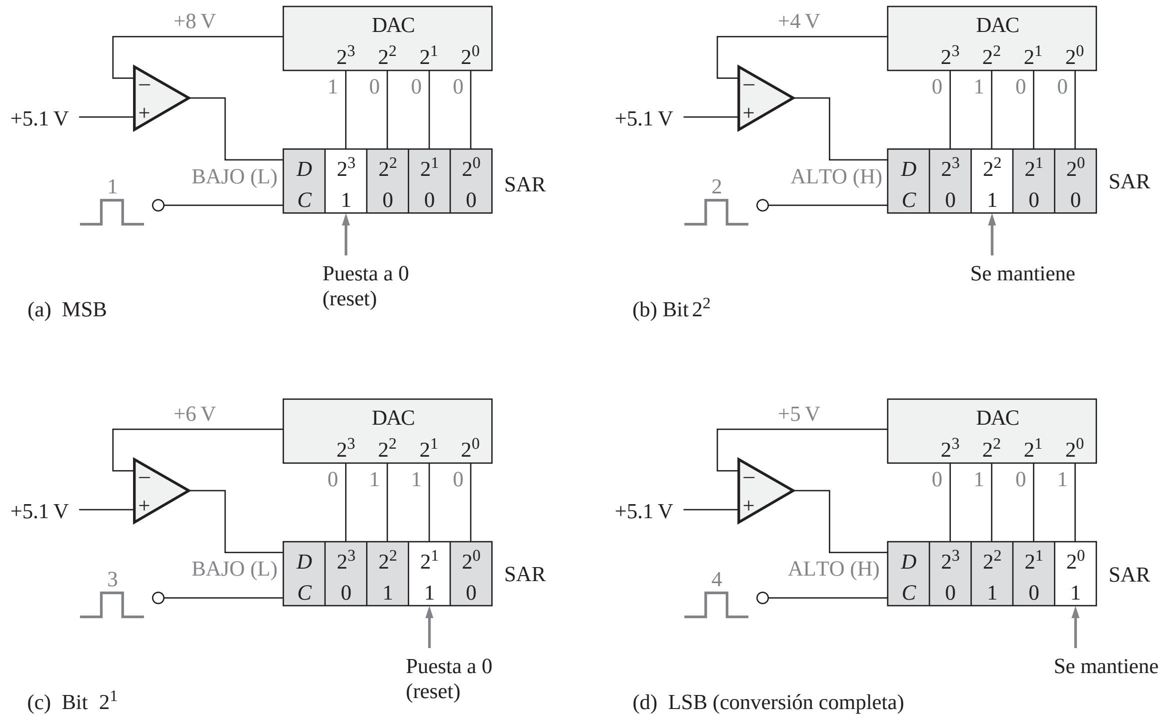
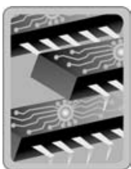


FIGURA 13.20 Ilustración del proceso de conversión por aproximaciones sucesivas.

Se han probado los cuatro bits y el ciclo de conversión ha sido completado. En este momento, el código binario almacenado en el registro es 0101, que es aproximadamente el valor binario de la entrada analógica de 5,1 V. Más bits producirán un resultado aún más preciso. A continuación, se inicia un nuevo ciclo de conversión y el proceso se repite. El SAR se borra al comienzo de cada nuevo ciclo.



EL CONVERTIDOR ANALÓGICO-DIGITAL ADC0804

El ADC0804 es un ejemplo de ADC por aproximaciones sucesivas. En la Figura 13.21 se presenta el diagrama de bloques. Este dispositivo funciona con una alimentación de +5V y tiene una resolución de ocho bits, con un tiempo de conversión de 100 μs. También dispone de un generador de reloj interno. Las salidas de datos triestado sirven para realizar la interfaz con el sistema de buses de un microprocesador.

El funcionamiento básico del dispositivo es el siguiente: el ADC0804 contiene el equivalente a una red DAC de 256 resistencias. La lógica de aproximaciones sucesivas secuencia la red para adaptar la tensión analógica de entrada diferencial ($V_{in+} - V_{in-}$) a una salida de la red resistiva. En primer lugar, se comprueba el MSB. Después de realizar ocho comparaciones (sesenta y cuatro períodos de reloj), un código binario de 8 bits se transfiere a los *latches* de salida y la salida de interrupción ($\overline{INTR}$) pasa a nivel BAJO. El dispositivo puede funcionar en modo de conversión libre (*free-running*), conectando la salida $\overline{INTR}$ a la entrada de escritura ($\overline{WR}$) y manteniendo la entrada de inicio de conversión, $\overline{CS}$, a nivel BAJO. Para garantizar una adecuada inicialización bajo todas las posibles condiciones, se requiere un nivel BAJO en la entrada $\overline{WR}$ durante el ciclo de

conexión de la alimentación. A partir de ahí, si se pone $\overline{CS}$ a nivel BAJO en cualquier instante, se interrumpirá el proceso de conversión.

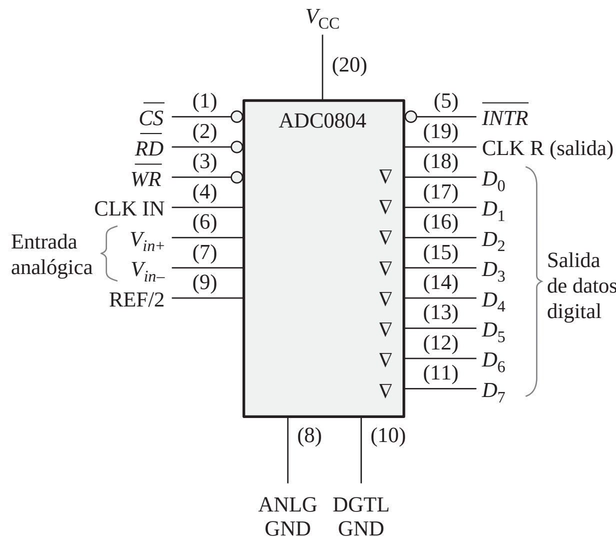


FIGURA 13.21 Convertidor analógico-digital ADC0804.

Cuando la entrada $\overline{WR}$ pasa a nivel BAJO, el registro de aproximaciones sucesivas (SAR) interno y el registro de desplazamiento de 8 bits se ponen a cero. Mientras, $\overline{CS}$ y $\overline{WR}$ permanezcan a nivel BAJO, el ADC permanecerá en estado de RESET. El período de conversión se inicia de uno a ocho períodos de reloj después de que $\overline{CS}$ o $\overline{WR}$ hagan una transición de nivel BAJO a nivel ALTO.

Cuando ambas entradas $\overline{CS}$ y $\overline{RD}$ están a nivel BAJO, el *latch* de salida triestado se habilita y el código de salida se aplica a las líneas D_0 a D_7. Cuando la entrada $\overline{CS}$ o la entrada $\overline{RD}$ pasan a nivel ALTO, las salidas D_0 a D_7 se desactivan.

Convertidor analógico-digital sigma-delta

Sigma-delta es un método muy extendido de conversión analógico-digital, particularmente cuando se utilizan señales de audio en el campo de las telecomunicaciones. El método está basado en la **modulación delta**, en la que se cuantifica la diferencia entre dos muestras sucesivas (es decir, se cuantifica el incremento o decremento sufrido por la señal); otros métodos utilizados en los convertidores ADC están basados en el valor absoluto de cada muestra. La modulación delta es un método de cuantificación de un bit.

La salida de un modulador delta es un flujo de datos de un único bit en el que el número relativo de 1s y 0s indica el nivel o amplitud de la señal de entrada. El número de 1s a lo largo de un cierto número de ciclos de reloj establece la amplitud de la señal durante dicho intervalo. Un número máximo de 1s corresponde a la tensión de entrada positiva más alta. Un número de 1s igual a la mitad del máximo se corresponde con una tensión de entrada igual a cero. Si no hay ningún 1 (si todos son 0s), lo que tenemos es una tensión de entrada negativa de máxima amplitud. Esto se ilustra de manera simplificada en la Figura 13.22. Por ejemplo, suponga que hay 4096 1s durante el intervalo en el que la señal de entrada presenta un máximo positivo. Puesto que el cero es el punto medio del rango dinámico de la señal de entrada, aparecerán 2048 1s durante

el intervalo en que esa señal es cero. Cuando la señal de entrada presenta un máximo negativo no habrá ningún 1 durante el intervalo. Para los niveles de señal intermedios, el número de 1s es proporcional al nivel de la señal.

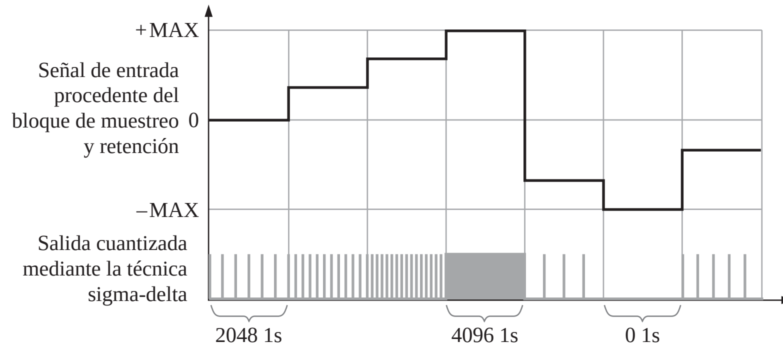


FIGURA 13.22 Ilustración simplificada de la conversión analógico-digital sigma-delta.

Diagrama de bloque funcional de un ADC sigma-delta El diagrama de bloques básico de la Figura 13.23 permite implementar el proceso de conversión ilustrado en la Figura 13.22. La señal analógica de entrada y la señal analógica correspondiente al flujo de bits cuantizado resultante de la conversión en el DAC, dentro del bucle de realimentación, se aplican al punto de suma (Σ). La señal diferencia (Δ) saliente de Σ se integra y el ADC de 1 bit incrementa o decrementa el número de 1s dependiendo de la señal diferencia. Esta acción trata de que la señal cuantizada realimentada se asemeje lo más posible a la señal analógica entrante. El cuantizador de 1 bit es esencialmente un comparador, seguido de un *latch*.

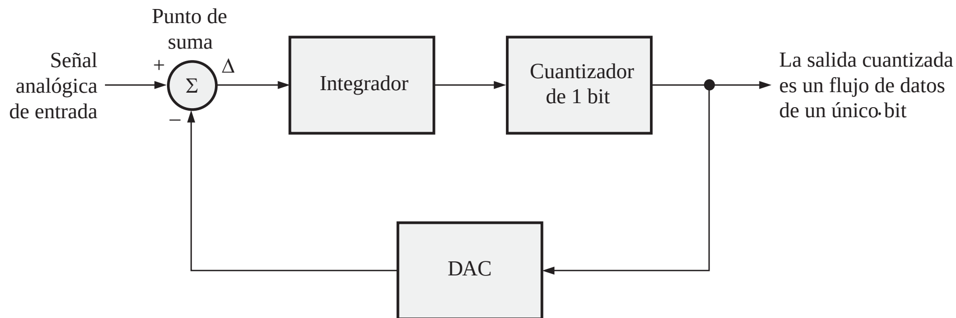


FIGURA 13.23 Diagrama de bloques funcional parcial de un ADC sigma-delta.

Para completar el proceso de conversión sigma-delta utilizando una técnica concreta, el flujo de datos de un único bit se convierte en una serie de códigos binarios, como se muestra en la Figura 13.24. El contador cuenta los 1s en el flujo de datos cuantizado durante cada uno de los sucesivos intervalos. El código almacenado en el contador representa entonces la amplitud de la señal analógica de entrada para cada intervalo. Estos códigos se enclavan en el *latch* para su almacenamiento temporal. La salida del *latch* es una serie de códigos de *n* bits, que representan de manera completa la señal analógica.

Otra de las posibles técnicas utiliza un filtro digital de decimación para generar la salida en lugar de emplear un contador y un *latch*, sin embargo, este tema queda fuera del alcance de este capítulo.

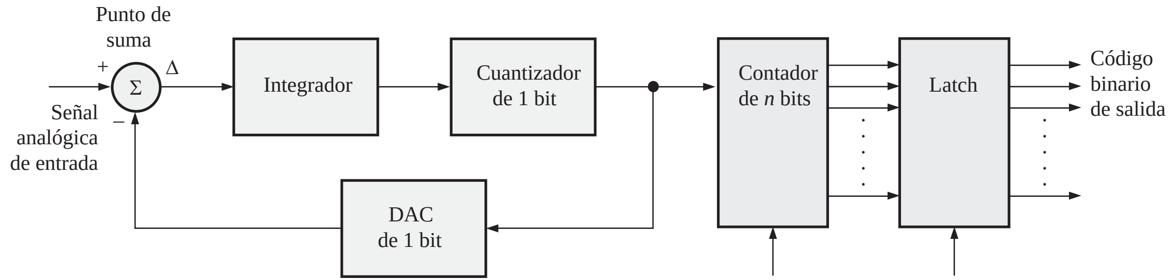


FIGURA 13.24 Un tipo de convertidor ADC sigma-delta.

Prueba de convertidores analógico-digitales

En la Figura 13.25 se presenta un método para la prueba de convertidores ADC. Se utiliza un DAC como parte de la configuración de pruebas, con el fin de convertir de nuevo a forma analógica la salida del ADC, para compararla con la entrada de prueba.

Aplicamos una entrada de prueba en forma de rampa lineal al ADC. A continuación, la secuencia de salida binaria se aplica al DAC de la unidad de prueba y se convierte en una rampa escalonada. Por último, se comparan las rampas de entrada y de salida para ver si existe alguna desviación.

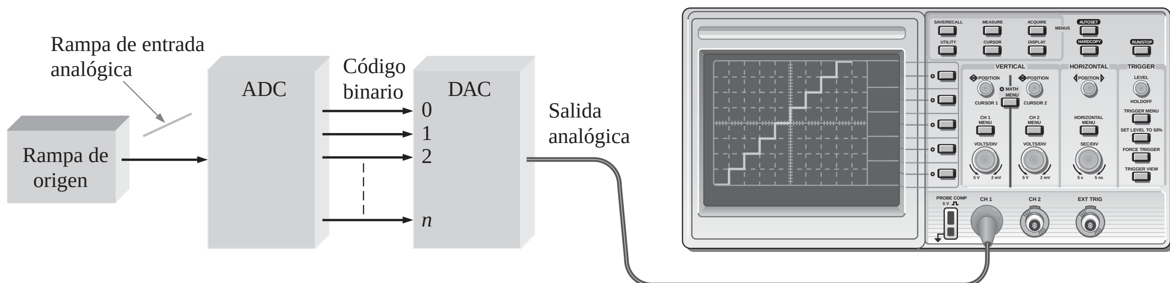


FIGURA 13.25 Método de prueba de un convertidor ADC.

Errores en la conversión analógico-digital

De nuevo utilizamos una conversión de 4 bits para ilustrar los principios. Supongamos que la entrada de prueba es una rampa lineal ideal.

Código ausente La salida en escalera de la Figura 13.26(a) indica que el código binario 1001 no aparece en la salida del ADC. Observe que el valor 1000 permanece durante dos intervalos y que la salida salta al valor 1010.

En el convertidor ADC flash, por ejemplo, un fallo en un comparador puede ser la causa del error de omisión de código.

Códigos incorrectos La salida en escalera de la Figura 13.26(b) indica que varias de las palabras del código binario que salen del ADC son incorrectas. Un análisis indica que, en este caso concreto, la línea del bit 2^1 permanece en estado BAJO (0).

Offset En la Figura 13.26(c) se presenta la condición de *offset*. En esta situación, el ADC interpreta que la tensión analógica de entrada es mayor que su valor real.

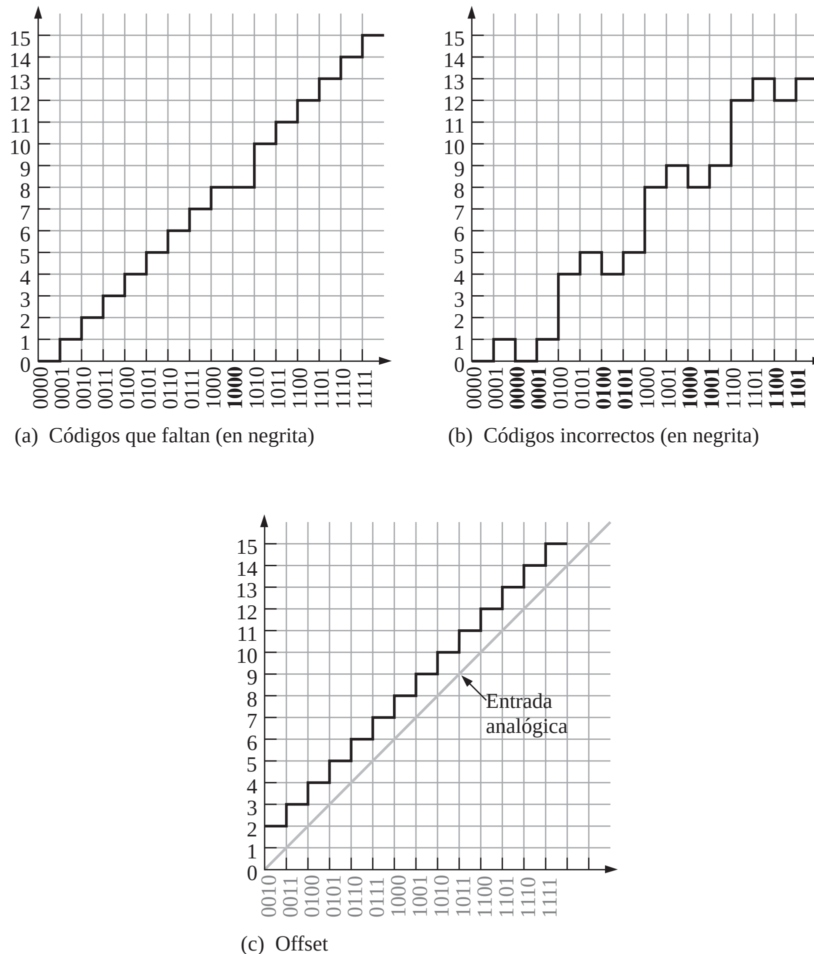


FIGURA 13.26 Ilustración de los errores de la conversión analógico-digital.

EJEMPLO 13.2

En la Figura 13.27(a) se presenta un ADC flash de 4 bits, el cual se prueba con una configuración como la mostrada en la Figura 13.25. En la Figura 13.27(b) se muestra la señal analógica de salida reconstruida. Identificar el problema y el fallo más probable.

Solución El código binario 0011 está ausente en la salida del ADC, como indica el escalón que falta. Muy probablemente, la salida del comparador 3 permanece en su estado inactivo (nivel BAJO).

Problema relacionado Reconstruir la señal de salida con un montaje como el de la Figura 13.25, si el comparador 8 del ADC de la Figura 13.27(a) permanece en el estado de salida ALTO.

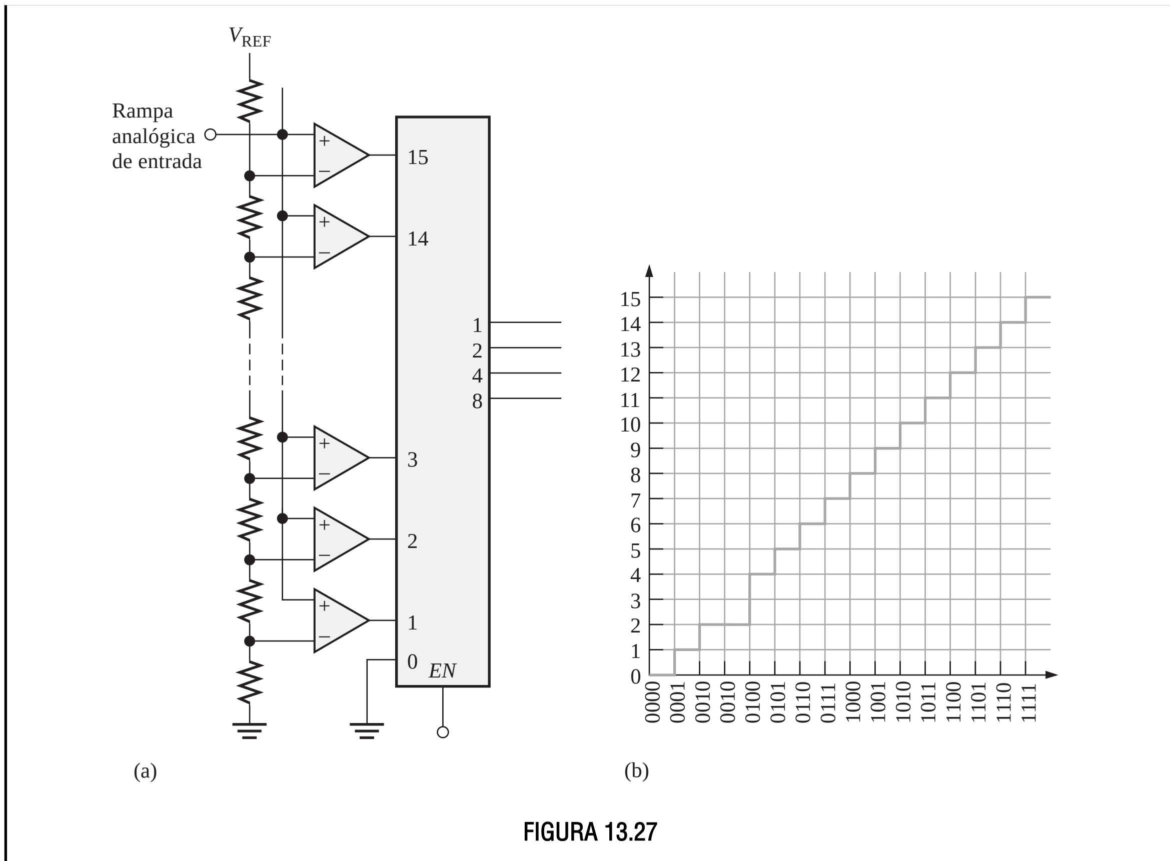


FIGURA 13.27

REVISIÓN DE LA SECCIÓN 13.3

1. ¿Cuál es el método más rápido de conversión analógico-digital?
2. ¿Qué método de conversión analógico-digital genera un flujo de datos de un único bit?
3. ¿Tiene un tiempo fijo de conversión el convertidor de aproximaciones sucesivas?
4. Citar dos tipos de errores de salida en un convertidor ADC.

13.4 PROCESADOR DIGITAL DE SEÑAL (DSP)

Esencialmente, un procesador digital de señal (DSP, *Digital Signal Processor*) es un tipo especial de microprocesador capaz de procesar los datos en tiempo real. Sus aplicaciones se centran en el procesamiento de datos digitales que representan señales analógicas. Un DSP, como un microprocesador, tiene una unidad central de proceso (CPU) y una unidad de memoria, además de diversas funciones de interfaz. Cada vez que se utiliza un teléfono móvil, estamos empleando un DSP, y éste es sólo un ejemplo de las muchas aplicaciones de este tipo de circuito.

Al finalizar esta sección el lector será capaz de:

■ Explicar los conceptos fundamentales en que se basa un DSP. ■ Indicar algunas de las aplicaciones de los DSP. ■ Describir las funciones básicas de un DSP en un teléfono móvil. ■ Describir la familia de procesadores DSP de la serie TMS320C6000.

El procesador digital de señal (DSP) es el corazón de los sistemas de procesamiento de señales digitales. Este circuito toma como entrada los datos proporcionados por un ADC y genera una salida que se aplica a un DAC, como se muestra en la Figura 13.28. Como hemos visto anteriormente, el ADC transforma una señal analógica en una serie de datos con codificación binaria que se entrega al DSP para su procesamiento. Después de procesados por el DSP, los datos pasan a un DAC para volver a ser convertidos a su forma analógica.

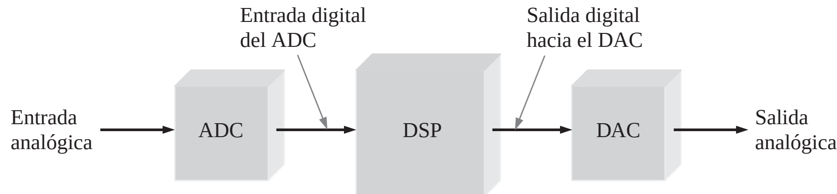


FIGURA 13.28 El DSP tiene una entrada digital y produce una salida digital.

Programación de un DSP

Los DSP se programan normalmente en lenguaje ensamblador o en C. Puesto que los programas escritos en ensamblador suelen ejecutarse más rápido y dado que la velocidad resulta crítica en la mayoría de las aplicaciones de procesamiento digital de la señal, el lenguaje ensamblador se emplea mucho más con los DSP que con los microprocesadores de propósito general. Asimismo, los programas para DSP suelen ser mucho más cortos que los programas tradicionales para microprocesador dado que sus aplicaciones son muy especializadas, por lo que hacen un gran uso de técnicas de redundancia. En general, el conjunto de instrucciones de un DSP tiende a ser más pequeño que el de un microprocesador.

Aplicaciones de los DSP

Un DSP, a diferencia de un microprocesador de propósito general, normalmente tiene que procesar los datos en *tiempo real*; es decir, a medida que van siendo generados. Muchas de las aplicaciones en las que se emplea un DSP no toleran que exista ningún tipo de retardo perceptible, lo que requiere que el DSP sea extremadamente rápido. Además de los teléfonos móviles, los DSP se utilizan en computadoras multimedia, cámaras de vídeo, reproductores de CD, unidades de disco duro, modems para radio digital y otras aplicaciones, con el fin de mejorar la calidad de las señales. Asimismo, los procesadores de tipo DSP van haciéndose cada vez más comunes en aplicaciones de televisión.

Una aplicación de gran importancia de un DSP es la compresión y descompresión de señales. En los sistemas de tipo CD, por ejemplo, la música almacenada en el CD está en formato comprimido para poder ahorrar espacio de almacenamiento. Por tanto, es necesario descomprimir esos datos para poder reproducir el sonido. Asimismo, también se emplean técnicas de compresión de señales en los teléfonos móviles, para poder gestionar un mayor número de llamadas en cada celda de la red. Otras áreas en las que los DSP tienen una gran importancia son las siguientes:

Telecomunicaciones El campo de las telecomunicaciones implica transferir todo tipo de informaciones de un lugar a otro, incluyendo conversaciones telefónicas, señales de televisión y datos digitales. Entre otras funciones el DSP facilita la multiplexación de muchas señales en un mismo canal de transmisión, porque la información en formato digital resulta relativamente sencilla de multiplexar y demultiplexar.

En el extremo transmisor de un sistema de telecomunicaciones, los DSP se emplean para comprimir las señales de voz digitalizadas, con el fin de ahorrar ancho de banda. La compresión es el proceso de reducir la frecuencia de los datos. Generalmente, las señales de voz se convierten a formato digital utilizando 8.000

muestras por segundo, valor éste que está basado en una frecuencia de Nyquist de 4 kHz. Si utilizamos 8 bits para codificar cada muestra, la frecuencia de los datos será de 64 kbits/s. En general, si reducimos (comprimimos) esa frecuencia de los datos de 64 kbits/s a 32 kbits/s, no se produce una pérdida significativa en la calidad del sonido. Cuando comprimimos los datos a 8 kbits/s, la calidad del sonido se reduce de manera perceptible. Cuando comprimimos los datos hasta un mínimo de 2 kbits/s, el sonido sufre una gran distorsión, pero sigue siendo utilizable en algunas aplicaciones en las que sólo importa poder reconocer las palabras pronunciadas, sin que la calidad del sonido sea relevante. En el extremo receptor de un sistema de telecomunicaciones, el DSP descomprime los datos para restaurar la señal a su forma original.

Los ecos que representan un problema en muchas conexiones telefónicas de larga distancia, se producen cuando una parte de la señal vocal es devuelta hacia el origen con un cierto retardo. En las distancias cortas, este retardo apenas resulta perceptible, pero a medida que se incrementa la distancia entre el transmisor y el receptor, el retardo del eco también se incrementa. Los DSP se emplean para cancelar de manera efectiva ese molesto eco, con lo que puede obtenerse una señal de voz clara y sin perturbaciones.

Procesamiento de señales musicales Los DSP se utilizan en la industria discográfica para proporcionar funciones de filtrado, de suma y resta de señales y de edición de sonido, durante la preparación y la grabación de señales musicales. Asimismo, otra aplicación de los DSP consiste en añadir efectos artificiales de eco y reverberación, que suelen verse minimizados por la acústica de los estudios de grabación, con el fin de simular entornos ideales de audición, desde salas de concierto a pequeñas habitaciones.

Generación y reconocimiento de voz Los DSP se emplean en la generación y reconocimiento de voz para mejorar la calidad de la comunicación hombre/máquina. El método que más comúnmente se utiliza para generar voz sintetizada en una computadora son las grabaciones digitales. Con la grabación digital, la voz humana se digitaliza y almacena usualmente en formato comprimido. Durante la reproducción, esos datos de voz almacenados se descomprimen y se devuelven a su formato analógico original. Aproximadamente, podemos almacenar una hora de información vocal utilizando aproximadamente unos 3 MB de memoria.

El reconocimiento de voz es mucho más difícil de llevar a cabo que la síntesis de voz. Incluso con las computadoras actuales, el reconocimiento de voz está muy limitado y, con unas pocas excepciones, los resultados han tenido un éxito sólo moderado. Los DSP se utilizan para aislar y analizar cada palabra de la señal de voz entrante. En cada palabra se identifican ciertos parámetros que se los compara con ejemplos previos de la señal vocal, con el fin de producir una correspondencia lo más exacta posible. La mayoría de los sistemas sólo pueden reconocer unos cuantos cientos de palabras, como mucho. Asimismo, hace falta emplear pausas significativas entre las distintas palabras y es también preciso “entrenar” al sistema para que se adapte a la voz de cada persona concreta. El reconocimiento del habla es un área en la que se están desarrollando enormes esfuerzos de investigación y que terminará por poder utilizarse en muchas aplicaciones comerciales.

Radar En las aplicaciones de radar, los DSP permiten realizar una determinación más precisa de las distancias usando técnicas de compresión de datos. También permiten reducir el ruido mediante técnicas de filtrado, con lo que el alcance del sistema se incrementa, así como optimizar la capacidad del sistema de radar para identificar tipos específicos de objetivos. Los DSP también se usan de forma similar en los sistemas de sonar.



NOTAS INFORMÁTICAS

Las tarjetas de sonido utilizadas en las computadoras utilizan un ADC para convertir en una señal digital el sonido procedente de un micrófono, de un reproductor de discos CD de audio o de algún otro tipo de fuente. El ADC envía la señal digital a un procesador digital de señal (DSP). Basándose en las instrucciones contenidas en una memoria ROM, una de las funciones del DSP consiste en comprimir la señal digital para que ésta ocupe un menor espacio de almacenamiento. El DSP envía entonces los datos comprimidos al procesador de la computadora que, a su vez, transmite esos datos a una unidad de disco duro o de CD-ROM para su almacenamiento. Para reproducir un sonido grabado, el procesador extrae los datos almacenados y los envía al DSP, que los descomprime y se los entrega a un DAC. La salida del DAC, que es una reproducción de la señal sonora original, se aplica a los altavoces.

Procesamiento de imágenes Los DSP se emplean en numerosas aplicaciones de procesamiento de imágenes, como la tomografía por computadora, la obtención de imágenes por resonancia magnética, dos técnicas ampliamente utilizadas en el campo de la medicina para poder explorar el interior del cuerpo humano. En la tomografía por computadora se hace pasar una serie de rayos X a través de una sección del cuerpo desde múltiples direcciones. Las señales resultantes se convierten a forma digital y se almacenan, utilizándose la información almacenada para generar imágenes calculadas que parecen cortes del cuerpo humano y que muestran un grado considerable de detalle, permitiendo así un mejor diagnóstico.

En lugar de rayos X, la resonancia magnética utiliza campos magnéticos y ondas de radio para sondear el interior del cuerpo humano. La resonancia magnética también genera imágenes, al igual que la tomografía por computadora, y permite una excelente discriminación entre los diferentes tipos de tejidos, así como la obtención de información, como por ejemplo el flujo de sangre a través de las arterias. La tecnología MRI depende enteramente de las técnicas de procesamiento digital de la señal.

En aquellas aplicaciones como los videoteléfonos, la televisión digital y otros sistemas de tratamiento de imágenes en movimiento, los DSP utilizan técnicas de compresión de imágenes para reducir el número de bits necesarios, haciendo que estos sistemas sean comercialmente factibles.

Filtrado Los DSP se utilizan comúnmente para implementar filtros digitales con el objetivo de separar señales que hayan sido combinadas con otras señales o que se hayan visto afectadas por interferencias y ruidos, así como para restaurar señales que hayan sufrido algún tipo de distorsión. Aunque los filtros analógicos resultan perfectamente adecuados en algunas aplicaciones, los filtros digitales suelen ser muy superiores en términos de las prestaciones que pueden obtenerse. Una de las desventajas de los filtros digitales es que el tiempo de ejecución requerido genera un retardo entre el momento en que se aplica la señal analógica y el instante en que aparece la salida. Los filtros analógicos no presentan ningún problema de retardo, porque la respuesta aparece en la salida en cuanto se aplica la entrada. Asimismo, los filtros analógicos son más baratos que los filtros digitales. Con independencia de esto, las prestaciones globales de un filtro digital son muy superiores en muchas aplicaciones.

Utilización de un DSP en un teléfono móvil

El teléfono móvil es un ejemplo de cómo puede utilizarse un DSP. La Figura 13.29 muestra un diagrama simplificado de un teléfono móvil. El **codec** de voz (codec es la abreviatura de codificador/decodificador) contiene, entre otras funciones, el ADC y DAC necesarios para efectuar la conversión entre la señal analógica de voz y el formato digital de voz. Normalmente se emplea una conversión sigma-delta en la mayoría de los teléfonos móviles. Para la transmisión, la señal de voz del micrófono se convierte a forma digital en el codec mediante el ADC, después de lo cual pasa al DSP para su procesamiento. Desde el DSP, la señal digital va a la sección de radiofrecuencia (RF), donde se la modula y se la cambia a la frecuencia de radio utilizada en transmisión. La señal RF entrante que contiene los datos de voz es captada por la antena, demodulada y transformada en una señal digital. Después de la entrega al DSP para su procesamiento, tras lo cual esa señal digital pasa al codec para volver a ser convertida, mediante el DAC, en la señal de voz original. Por último, esa señal se amplifica y se entrega al altavoz.

Funciones realizadas por el DSP. En una aplicación de telefonía móvil, el DSP realiza numerosas funciones para mejorar y facilitar la recepción y transmisión de las señales de voz. Algunas de esas funciones realizadas por el DSP son las siguientes:

- *Compresión de voz.* La frecuencia de la señal digital de voz se reduce significativamente antes de la transmisión, para poder satisfacer los requisitos de ancho de banda.
- *Descompresión de voz*. La señal digital de voz recibida se devuelve a su frecuencia original, para poder reproducir apropiadamente la señal de voz analógica.

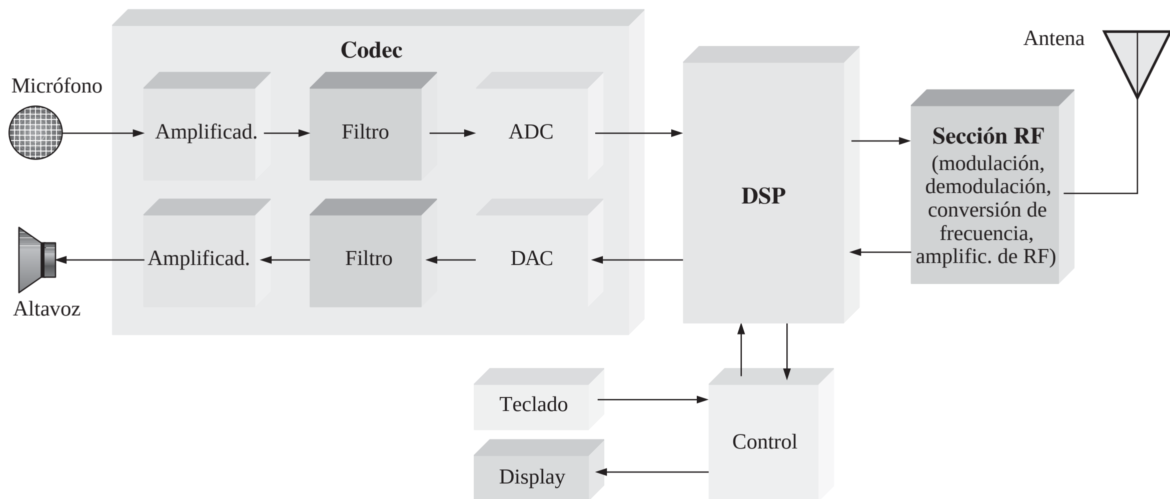


FIGURA 13.29 Diagrama de bloques simplificado de un teléfono móvil digital.

- *Gestión de protocolos*. El teléfono móvil se comunica con la estación base más próxima para poder determinar la ubicación del teléfono móvil, asignarle las franjas temporales y de frecuencia necesarias y gestionar la cesión a otras estaciones base cuando el teléfono entra en otra celda.
- *Detección y corrección de errores*. Durante la transmisión, se generan códigos de detección y corrección de errores. Durante la recepción, se detectan y corrigen los errores inducidos por los ruidos o interferencias existentes en el canal de RF.
- *Cifrado*. La señal de voz digital se convierte a otro formato para poder transmitirla con seguridad, efectuándose la conversión inversa a su forma original durante la recepción.

Arquitectura básica de un DSP

Como hemos mencionado anteriormente, un DSP es básicamente un microprocesador especializado que está optimizado en lo que respecta a su velocidad, con el fin de poder procesar los datos en tiempo real. Muchos de los DSP están basados en lo que se denomina *arquitectura Harvard*, que consiste en una unidad central de proceso (CPU) y dos memorias, una para los datos y otra para el programa, como se muestra en la Figura 13.30.

Un DSP específico: la serie TMS320C6000

Hay muchas empresas fabricantes de DSP, entre las que se incluyen Texas Instruments, Motorola y Analog Devices. Hay disponibles DSP tanto para procesamiento de coma fija como de coma flotante. Recuerde del Capítulo 2 que estos dos métodos difieren en la forma en que se almacenan y manipulan los números. Todos los procesadores DSP de coma flotante pueden también procesar números en formato de coma fija. Los procesadores DSP de coma fija son más baratos que las versiones de coma flotante y pueden, generalmente, operar con mayor velocidad. Los detalles de la arquitectura de un DSP pueden variar de manera significativa, incluso dentro de una misma familia. Analizamos brevemente una serie concreta de DSP, como ejemplo de la forma en que se suele organizar un DSP.

Entre los ejemplos de los DSP disponibles en la serie TMS320C6000 tenemos el TMS320C62xx, el TMS320C64xx y el TMS320C67xx, que forman parte de la familia de dispositivos TMS320 de Texas Instruments. En la Figura 13.31 se muestra un diagrama de bloques general de estos dispositivos.

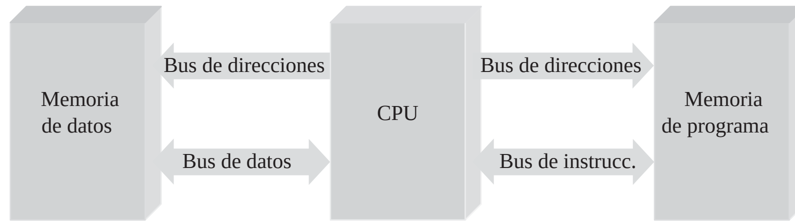


FIGURA 13.30 Muchos DSP utilizan la arquitectura Harvard (con dos memorias).

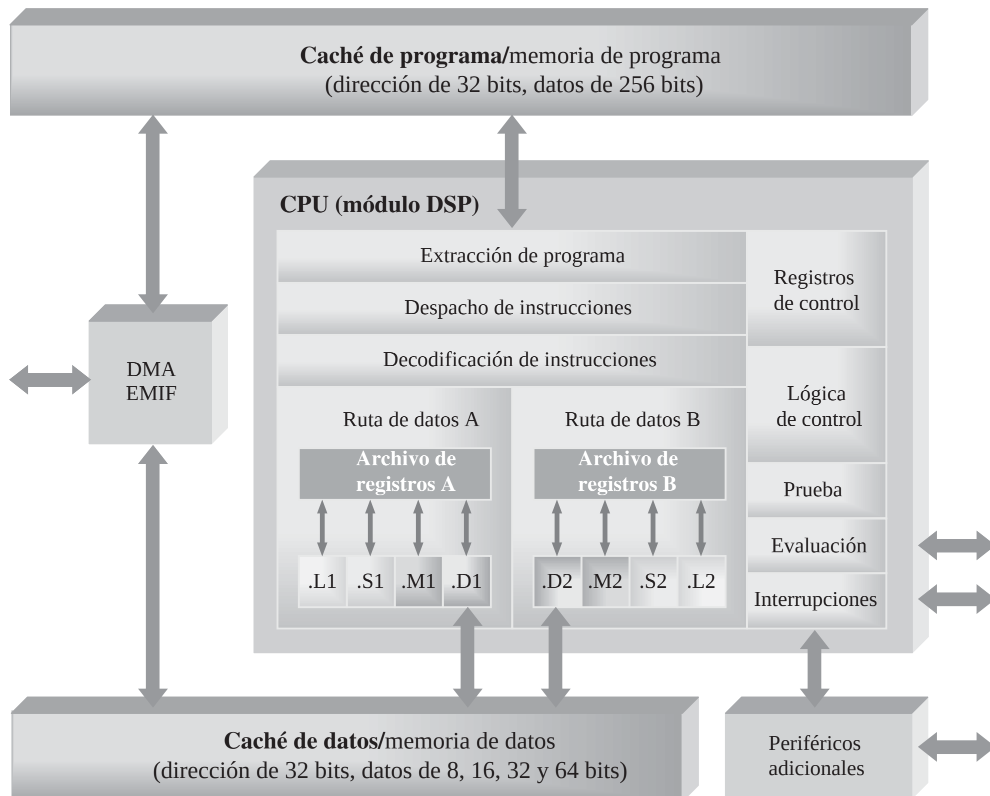


FIGURA 13.31 Diagrama de bloques general de la serie de procesadores DSP TMS320C6000.

Los DSP tienen una unidad central de proceso (CPU) también conocida como ***módulo DSP***, que contiene 64 registros de propósito general de 32 bits en el C64xx y 32 registros de propósito general de 32 bits en el C62xx y el C67xx. El C67xx puede realizar operaciones de coma flotante, mientras que el C62xx y el C64xx son procesadores de coma fija.

Cada DSP tiene ocho unidades funcionales que contienen dos multiplicadores de 16 bits y seis unidades aritmético-lógicas (ALU). Las prestaciones de los tres DSP de la serie C6000 en términos de ***MIPS*** (*Million Instructions Per Second*, millones de instrucciones por segundo), ***MFLOPS*** (*Million Floating-Point Operations Per Second*, millones de operaciones en coma flotante por segundo) y ***MMACS*** (*Million Multiply/Accumulates Per Second*, millones de multiplicación/acumulación por segundo) se muestran en la Tabla 13.3

DSP	Tipo	Aplicación	Velocidad de procesamiento	Velocidad de multiplicación/acumulación
C62xx	Coma fija	Propósito general	1200–2400 MIPS	300–600 MMACS
C64xx	Coma fija	Propósito especial	3200–4800 MIPS	1600–2400 MMACS
C67xx	Coma flotante	Propósito general	600–1000 MFLOPS	200–333 MMACS

TABLA 13.3 Prestaciones de los DSP de la serie TMS320C6000 para procesamiento de datos.

Rutas de datos en la CPU En la CPU, las secciones de extracción de programa, despacho de instrucciones y decodificación de instrucciones pueden proporcionar ocho instrucciones de 32 bits a las unidades funcionales en cada ciclo de reloj. La CPU está dividida en dos rutas de datos y el procesamiento de instrucciones tiene lugar tanto en la ruta de datos A como en la B. Cada ruta de datos contiene una mitad del conjunto de registros de propósito general (16 en el C62xx y el C67xx o 32 en el C64xx) y cuatro unidades funcionales. Los registros de control y la lógica de control se utilizan para configurar y controlar las diversas operaciones del procesador.

Unidades funcionales Cada ruta de datos tiene cuatro unidades funcionales. Las unidades M (etiquetadas como .M1 y .M2 en la Figura 13.31) son multiplicadores dedicados. Las unidades L (etiquetadas como .L1 y .L2) realizan operaciones aritméticas, lógicas y de carácter misceláneo. Las unidades S (etiquetadas como .S1 y .S2) realizan operaciones de comparación, operaciones de desplazamiento y operaciones aritméticas misceláneas. Las unidades D (etiquetadas como .D1 y .D2) realizan operaciones de carga, de almacenamiento y misceláneas.

Pipeline Una **pipeline** permite procesar simultáneamente múltiples instrucciones. Una operación *pipeline* consta de tres etapas a través de las cuales fluyen todas las instrucciones: *extracción*, *decodificación*, *ejecución*. En primer lugar, se extraen simultáneamente ocho instrucciones de la memoria de programa; esas instrucciones se decodifican a continuación y finalmente se ejecutan.

Durante la ***extracción***, las ocho instrucciones (que reciben el nombre de paquete) se extraen de la memoria en cuatro *fases*, como se muestra en la Figura 13.32.

- *Generación de la dirección del programa (PG).* La CPU genera la dirección de programa.
- *Envío de la dirección de programa (PS).* Se envía a la memoria la dirección de programa.
- *Espera de acceso de programa (PW).* Se produce una operación de lectura de memoria.
- *Recepción del paquete de programa extraído (PR).* La CPU recibe el paquete de instrucciones.

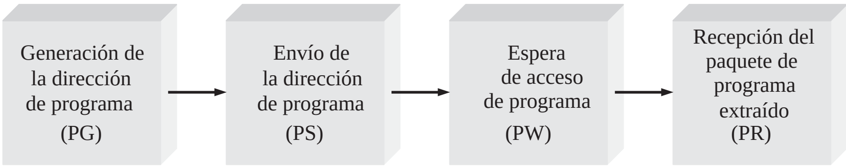


FIGURA 13.32 Las cuatro fases de extracción de una operación *pipeline*.

La etapa de ***decodificación*** de una operación *pipeline* está compuesta de dos fases, como se muestra en la Figura 13.33. En la fase de despacho de instrucciones (DP) es en la que se dividen los paquetes de instrucciones en paquetes de ejecución, que se asignan a las unidades funcionales apropiadas. En la fase de decodificación de instrucciones (DC) se decodifican las instrucciones recibidas.

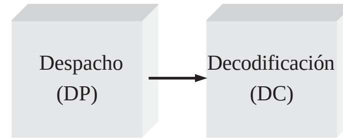


FIGURA 13.33 Las dos fases de decodificación de una operación *pipeline*.

En la etapa de ***ejecución*** de la operación *pipeline* se llevan a cabo las instrucciones proporcionadas por la etapa de decodificación. La etapa de ejecución tiene un máximo de cinco fases (E1 a E5), como se muestra en la Figura 13.34. No todas las instrucciones utilizan las cinco fases. El número de fases empleado durante la ejecución dependerá del tipo de instrucción. Parte de la ejecución de una instrucción requiere obtener datos de la memoria de datos.

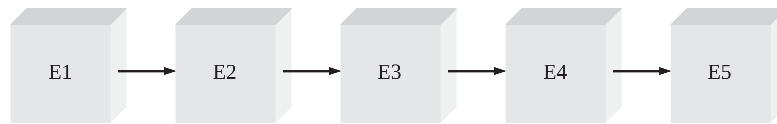


FIGURA 13.34 Las cinco fases de ejecución de una operación *pipeline*.

Interfaces y memoria interna de un DSP Como se puede ver en la Figura 13.31 existen dos memorias internas: una para los datos y otra para el programa. La memoria de programa está organizada en paquetes de 256 bits (ocho instrucciones de 32 bits) y tiene una capacidad de 64 kB. La memoria de datos también tiene una capacidad de 64 kB y se puede acceder a ella con longitudes de palabra de 8, 16, 32 o 64 bits, dependiendo del dispositivo específico de la serie. A ambas memorias internas se accede utilizando una dirección de 32 bits. Se utiliza un mecanismo DMA (acceso directo a memoria) para transferir los datos sin que estos tengan que pasar a través de la CPU. La interfaz EMIF (*External Memory Inteface*, interfaz de memoria externa) se utiliza para soportar memorias externas, en caso de que la aplicación lo requiera. Se proporcionan interfaces adicionales para los puertos de E/S serie y otros dispositivos externos.

Temporizadores Hay dos temporizadores de propósito general en el DSP que pueden utilizarse para sucesos temporizados, para tareas de recuento, para generación de pulsos, para generación de interrupciones de la CPU y para otras tareas.

Encapsulados Estos procesadores concretos están disponibles en encapsulados BGA (*Ball Grid Array*) de 352 pines, como se muestra en la Figura 13.35 y están implementados con tecnología CMOS.

REVISIÓN DE LA SECCIÓN 13.4

1. ¿Qué es la arquitectura Harvard?
2. ¿Qué es un módulo DSP?
3. Citar dos categorías de procesadores DSP de acuerdo con el tipo de los datos numéricos que pueden procesar.
4. ¿Cuáles son los dos tipos de memoria interna?
5. Definir los términos

   (a) MIPS **(b)** MFLOPS **(c)** MMACS.
6. Básicamente, ¿qué es lo que una *pipeline* permite conseguir?
7. Nombrar las tres etapas de una operación *pipeline.*
8. ¿Qué sucede durante la fase de extracción?

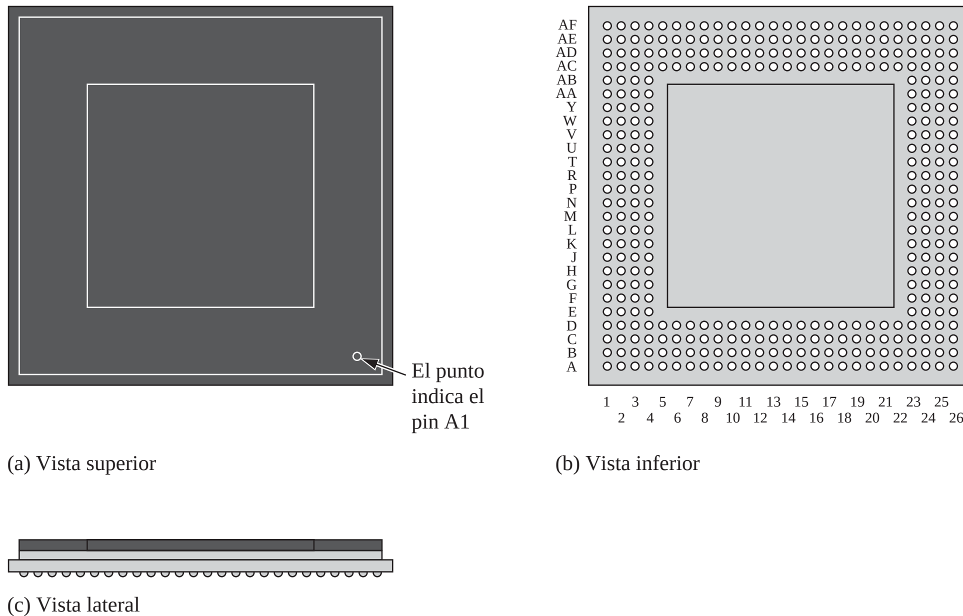


FIGURA 13.35 Un encapsulado BGA de 352 pines.

13.5 MÉTODOS DE CONVERSIÓN DIGITAL-ANALÓGICA

La conversión digital-analógica es una parte importante de los sistemas de procesamiento de señales digitales. Una vez procesados los datos digitales por el DSP, los convierte de nuevo a forma analógica. En esta sección vamos a examinar la teoría de operación de dos tipos básicos de convertidores digital-analógicos (DAC) y vamos a analizar sus correspondientes prestaciones.

Al finalizar esta sección el lector será capaz de:

■ Explicar la operación de un DAC con ponderación binaria. ■ Explicar la operación de un DAC en escalera $R/2R$. ■ Definir los conceptos de resolución, precisión, linealidad, monotonicidad y tiempo de asentamiento de un DAC. ■ Indicar cómo se prueba la no monotonicidad, la no linealidad diferencial, la existencia de una ganancia alta o baja o la existencia de errores de *offset* en un DAC.

Convertidor analógico-digital con ponderación binaria

Existe un método de conversión digital/analógica que utiliza una red resistiva en la que los valores de las resistencias representan los pesos binarios de los bits de entrada del código digital. La Figura 13.36 muestra un DAC de 4 bits de este tipo. Por cada una de las resistencias de entrada puede circular o no corriente, dependiendo del nivel de tensión de entrada. Si la tensión de entrada es cero (0 binario), la corriente también es cero. Si la tensión de entrada es un nivel ALTO (1 binario), la cantidad de corriente depende del valor de la resistencia de entrada y es diferente para cada una de las resistencias, como se indica en la figura.

Puesto que, prácticamente, no circula corriente por la entrada inversora (–) del amplificador operacional, la suma de todas las corrientes de entrada pasa a través de R_f. Como la entrada inversora está a 0 V (tierra virtual), la caída en R_f es igual a la tensión de salida, es decir, $V_{out} = I_f R_f$.

Los valores de las resistencias de entrada se seleccionan de modo que sean inversamente proporcionales a los pesos binarios de los correspondientes bits de entrada. La resistencia de menor valor (R) corresponde a

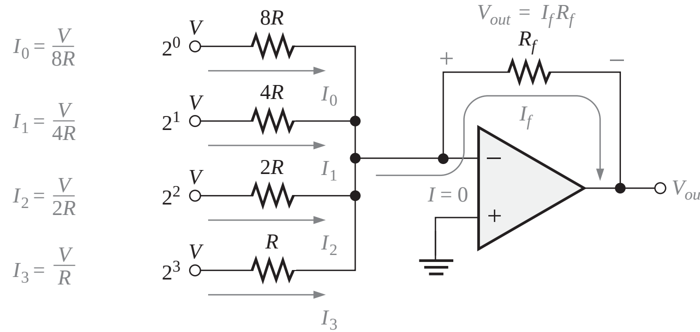


FIGURA 13.36 DAC con ponderación binaria de 4 bits.

la entrada ponderada más alta (2^3). Las restantes resistencias son múltiplos de R ($2R$, $4R$ y $8R$) y corresponden a los pesos binarios 2^2, 2^1 y 2^0, respectivamente. Las corrientes de entrada también son proporcionales a los pesos binarios. Luego la tensión de salida es proporcional a la suma de los pesos binarios, ya que es la suma de las corrientes de entrada por R_f.

Una de las desventajas de este tipo de DAC es el número de resistencias diferentes que utiliza y el hecho de que los niveles de tensión deben ser exactamente iguales en todas las entradas. Por ejemplo, un convertidor de 8 bits requiere ocho resistencias en el rango que va desde R hasta $128R$, en pasos ponderados. Este rango de resistencias requiere tolerancias de 1 entre 255 (menor del 0,5%) para convertir la entrada con precisión, lo que hace muy difícil fabricar este tipo de DAC en grandes cantidades.

EJEMPLO 13.3

Determinar la salida del DAC de la Figura 13.37(a), si se aplican a las entradas las formas de onda (que representan una secuencia de números de 4 bits) de la Figura 13.37(b). La entrada D_0 es el bit menos significativo (LSB).

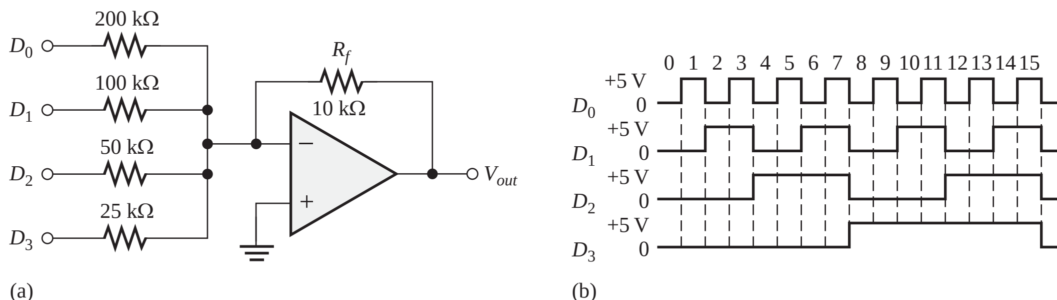


FIGURA 13.37

Solución En primer lugar, determinamos la corriente por cada una de las entradas ponderadas. Dado que la entrada inversora (−) del amplificador operacional está a 0 V (tierra virtual) y un 1 binario corresponde a +5 V, la corriente a través de cualquiera de las resistencias de entrada es de 5 V dividido entre el valor de la resistencia.

$$I_0 = \frac{5\,\text{V}}{200\ \text{k}\Omega} = 0{,}025\ \text{mA}$$

$$I_1 = \frac{5\,\text{V}}{100\ \text{k}\Omega} = 0{,}05\ \text{mA}$$

$$I_2 = \frac{5\,\text{V}}{50\ \text{k}\Omega} = 0{,}1\ \text{mA}$$

$$I_3 = \frac{5\,\text{V}}{25\ \text{k}\Omega} = 0{,}2\ \text{mA}$$

Por la entrada inversora del amplificador operacional casi no circula corriente, debido a su muy alta impedancia. Por tanto, suponemos que toda la corriente atraviesa la resistencia de realimentación R_f. Como un extremo de R_f está a 0V (tierra virtual), la caída en R_f es igual a la tensión de salida, que es negativa con respecto a la tierra virtual.

$$V_{out(D0)} = (10\ \text{k}\Omega)(-0{,}025\,\text{mA}) = -0{,}25\ \text{V}$$
$$V_{out(D1)} = (10\ \text{k}\Omega)(-0{,}05\,\text{mA}) = -0{,}5\ \text{V}$$
$$V_{out(D2)} = (10\ \text{k}\Omega)(-0{,}1\,\text{mA}) = -1\ \text{V}$$
$$V_{out(D3)} = (10\ \text{k}\Omega)(-0{,}2\,\text{mA}) = -2\ \text{V}$$

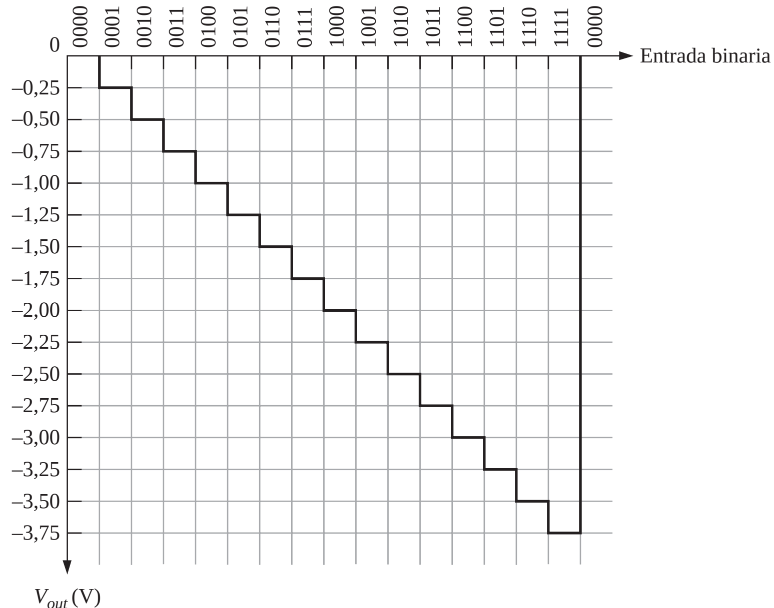


FIGURA 13.38 Salida del DAC de la Figura 13.37.

En la Figura 13.37(b), el primer código de entrada binario es 0000, que produce una tensión de salida de 0 V. El siguiente código de entrada es 0001, que da

lugar a una tensión de salida de –0,25 V. El siguiente código es 0010, que produce una tensión de salida de –0,5 V. El siguiente código es 0011, que da lugar a una tensión de salida de –0,25V + (– 0,5 V) = – 0,75V. Cada sucesivo código binario aumenta la tensión de salida en –0,25 V, por lo que, para esta secuencia binaria particular en las entradas, la salida es una forma de onda en escalera que va desde 0 V a –3,75 V, a escalones de –0,25 V. Esto se muestra en la Figura 13.38.

Problema relacionado Invertir las formas de onda de entrada del DAC de la Figura 13.37 (D_3 a D_0, D_2 a D_1, D_1 a D_2, D_0 a D_3) y determinar la salida.

Convertidor digital-analógico en escalera R/2R

Otro método para realizar la conversión digital-analógica es utilizar la red escalonada *R/2R*, como muestra la Figura 13.39 para el caso de cuatro bits. Este método resuelve uno de los problemas del DAC con ponderación binaria, ya que sólo requiere dos valores de resistencia.

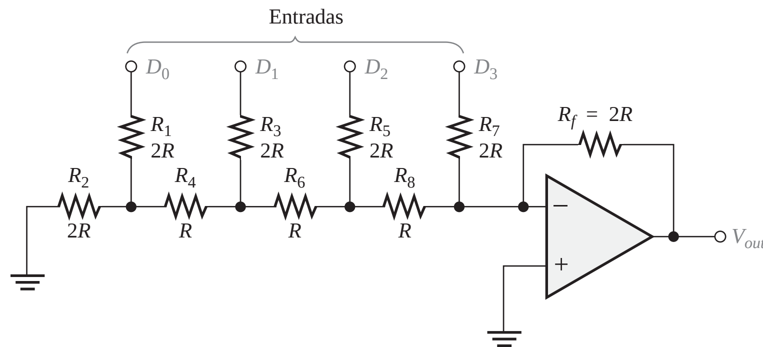


FIGURA 13.39 DAC en escalera R/2R.

Comencemos asumiendo que la entrada D_3 está a nivel ALTO (+5 V) y las demás a nivel BAJO (tierra, 0 V). Esta condición representa el número binario 1000. Un análisis del circuito demostraría que este circuito es equivalente al mostrado en la Figura 13.40(a). A través de la resistencia equivalente 2*R* prácticamente no circula corriente, ya que la entrada inversora está a tierra virtual. Luego toda la corriente (I = 5 V/2R) que circula a través de R_7 pasa también por R_f, y la tensión de salida es –5 V. El amplificador operacional mantiene la entrada inversora (–) a casi cero voltios (≈ 0V) debido a la realimentación negativa. Por tanto, toda la corriente pasa a través de R_f en lugar de por la entrada inversora.

La Figura 13.40(b) muestra el circuito equivalente cuando la entrada D_2 está a +5 V y las demás están a tierra. Esta condición representa 0100. Si se aplica el equivalente de Thevenin* mirando desde R_8 , se obtienen 2,5 V en serie con *R*, como se indica. Esto da lugar a una corriente a través de R_f de I = 2,5 V/2R, lo que determina una tensión de salida de –2,5 V. No olvide que no circula corriente por la entrada inversora ni por la resistencia equivalente conectada a tierra, ya que caen 0 V en ella debido a la tierra virtual.

* El teorema de Thevenin establece que cualquier circuito se puede reducir a una fuente de tensión equivalente en serie con una resistencia equivalente.

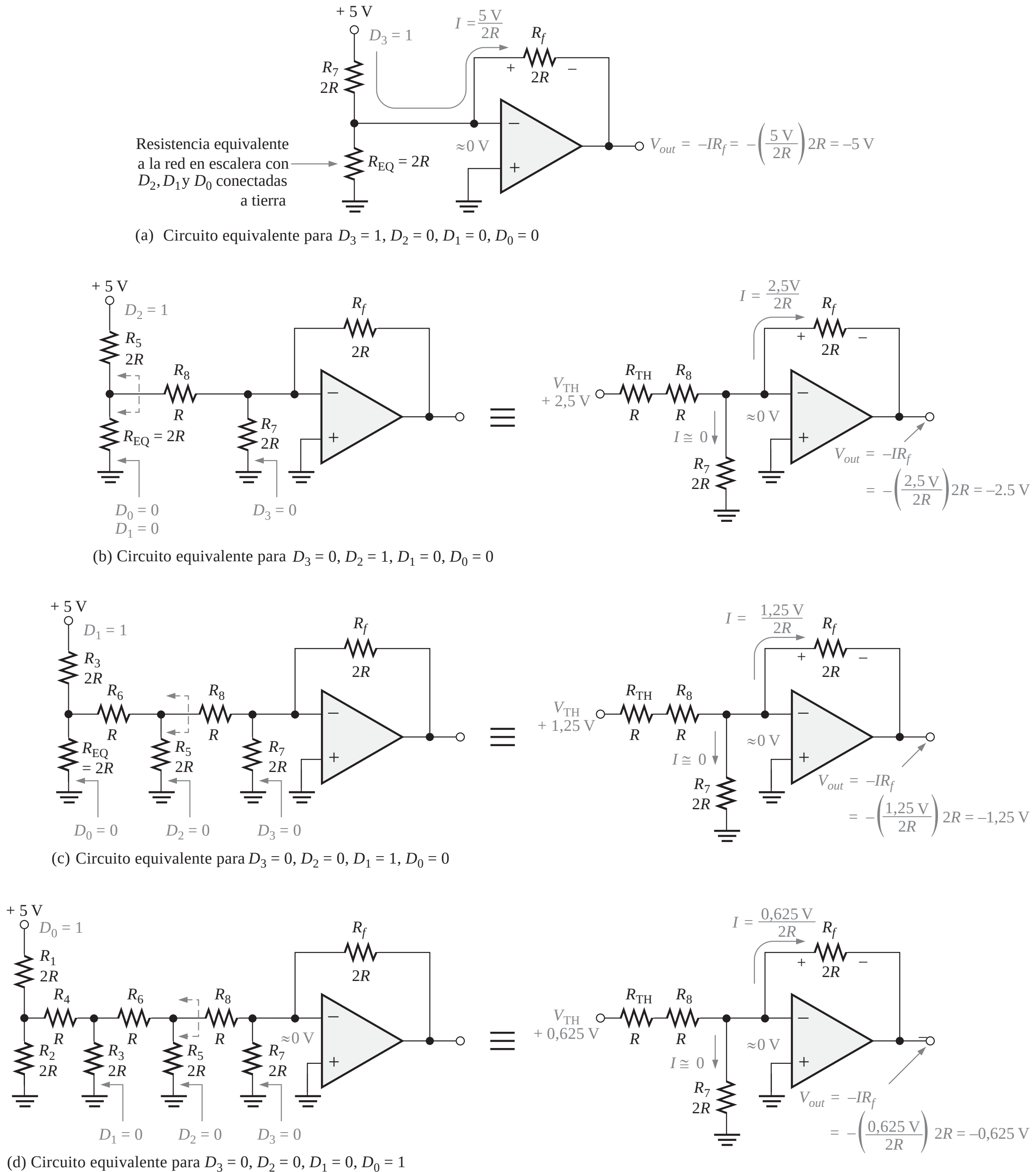


FIGURA 13.40 Análisis del DAC en escalera R/2R.

La Figura 13.40(c) muestra el circuito equivalente cuando la entrada D_1 está a + 5V y las demás están a tierra. Esta condición representa el código 0010. De nuevo, aplicando el teorema de Thevenin al circuito a la

izquierda de R_8, se obtienen 1,25 V en serie con *R,* como se indica. Esto da lugar a una corriente a través de R_f de I = 1,25 V/2R, lo que genera una tensión de salida de −1,25 V.

En la parte (d) de la Figura 3.40 se representa el circuito equivalente para el caso en que D_0 está a + 5V y las demás entradas están a tierra. Esta condición representa el código 0001. Aplicando el teorema de Thevenin desde R_8, se obtienen 0,625 V en serie con *R,* como se indica. Esto da lugar a una corriente a través de R_f de I = 0,625 V/2R, lo que genera una tensión de salida de −0,625 V.

Observe que cada entrada de menor peso sucesiva produce una tensión de salida que es la mitad de la anterior, por lo que la tensión de salida es proporcional al peso binario de los bits de entrada.

Características de funcionamiento de los convertidores analógico-digitales

Las prestaciones básicas de un DAC incluyen la resolución, precisión, linealidad, monotonicidad y tiempo de establecimiento, cada una de las cuales vamos a ver a continuación.

- *Resolución.* La resolución de un DAC es el recíproco del número de escalones discretos en la salida. Por supuesto, depende del número de bits de entrada. Por ejemplo, un DAC de 4 bits tiene una resolución de 1 entre $2^4 - 1$ (uno entre quince). Si se expresa como un porcentaje, será (1/15)100 = 6,67%. El número total de escalones discretos es igual a $2^n - 1$, donde n es el número de bits. La resolución también puede expresarse como el número de bits que se convierten.
- *Precisión.* La precisión es una comparación entre la salida real de un DAC y la salida esperada. Se expresa como un porcentaje de la tensión de salida a fondo de escala o máxima. Por ejemplo, si la salida a fondo de escala de un convertidor es 10 V y la precisión es ±0,1%, entonces el error máximo para cualquier tensión de salida es (10 V) (0,001) = 10 mV. Idealmente, la precisión debería ser, como mucho, ±1/2 del bit menos significativo. Para un convertidor de 8 bits, el bit menos significativo es el 0,39% del valor a fondo de escala. La precisión debería ser aproximadamente del ±0,2%.
- *Linealidad.* Un error lineal es una desviación de la salida ideal (una línea recta) del DAC. Un caso especial es el error de *offset*, que es la tensión de salida cuando los bits de entrada son todos cero.
- *Monotonicidad.* Un DAC es **monotónico** si no produce escalones invertidos cuando se le aplica secuencialmente su rango completo de bits de entrada.
- *Tiempo de asentamiento.* Normalmente, el tiempo de asentamiento se define como el tiempo que tarda un DAC en quedar dentro de ±1/2 LSB del valor final, cuando se produce un cambio en el código de entrada.

EJEMPLO 13.4

Determinar la resolución, expresada en porcentaje de **(a)** DAC de 8 bits. **(b)** DAC de 12 bits.

Solución **(a)** Para el convertidor de 8 bits,

$$\frac{1}{2^8 - 1} \times 100 = \frac{1}{255} \times 100 = \mathbf{0,392}\%$$

(b) Para el convertidor de 12 bits,

$$\frac{1}{2^{12} - 1} \times 100 = \frac{1}{4095} \times 100 = \mathbf{0,0244}\%$$

Problema relacionado Calcular la resolución de un DAC de 16 bits.

Prueba de un convertidor digital-analógico

La Figura 13.41 ilustra la manera de probar un DAC. Con este método básico se aplica una secuencia de códigos binarios a las entradas y se observa la salida resultante. La secuencia de códigos binarios cubre el rango completo de valores entre 0 y $2^n - 1$ en orden ascendente, donde n es el número de bits.

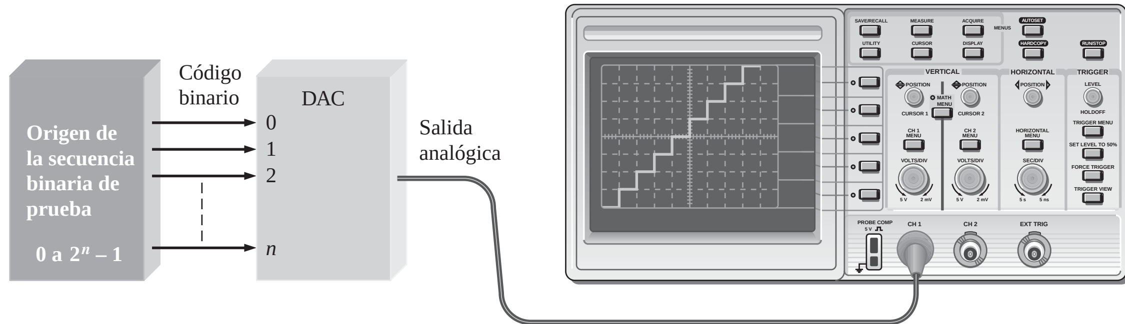


FIGURA 13.41 Configuración básica de pruebas para un DAC.

La salida ideal es una línea recta escalonada, tal como se indica en la figura. A medida que se incrementa el número de bits del código binario, mejora la resolución de conversión. En otras palabras, se incrementa el número de pasos discretos y la salida se aproxima a una rampa lineal.

Errores de la conversión digital-analógica

En la Figura 13.42 se muestran diversos errores potenciales de la conversión digital-analógica. En la figura se utiliza una conversión de cuatro bits, con el fin de ilustrar estos errores. Una conversión de 4 bits produce quince escalones discretos. Cada una de las gráficas de la figura incluye la rampa en escalera ideal, para compararla con las salidas que presentan fallos.

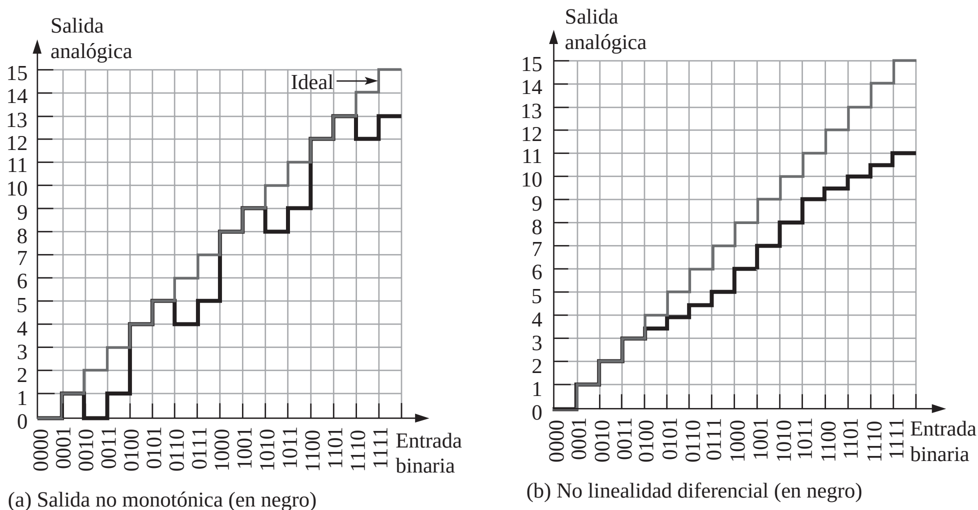


FIGURA 13.42 Ilustración de diversos errores de la conversión digital-analógica. (*Continúa*)

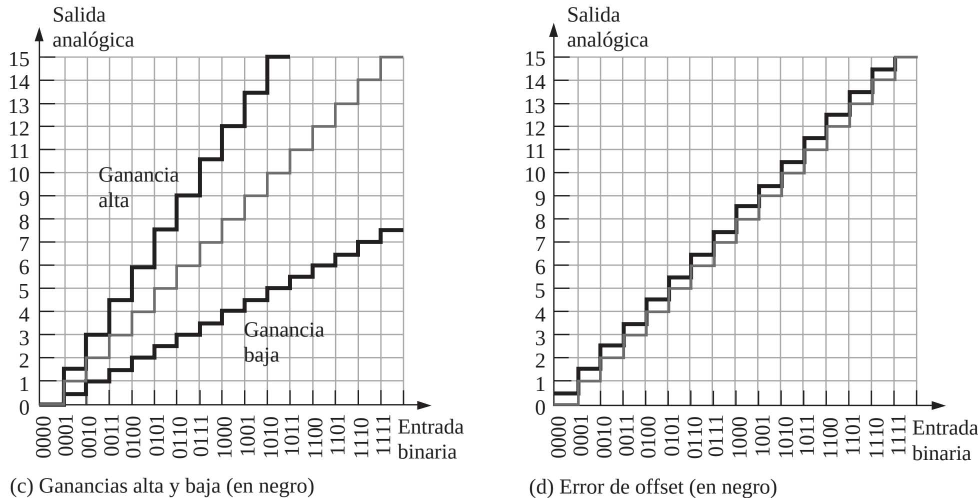


FIGURA 13.42 Ilustración de diversos errores de la conversión digital-analógica. (*Continuación*).

No monotonicidad Los escalones en sentido inverso de la Figura 13.42(a) indican un funcionamiento no monotónico, que es una forma de no linealidad. En este caso particular, el error se produce porque el bit 2^1 del código binario se interpreta como un 0 constante. Es decir, un cortocircuito hace que la línea de entrada del bit permanezca a nivel BAJO.

No linealidad diferencial La Figura 13.42(b) ilustra la no linealidad diferencial, en la que la amplitud del escalón es menor de lo que debería ser para ciertos códigos de entrada. Esta salida en concreto podría ser producida por el peso insuficiente del bit 2^2, debido a un fallo de la resistencia de entrada. También podría haber escalones con amplitudes mayores que la normal, si un peso en particular fuera mayor de lo que debería ser.

Baja y alta ganancia En la Figura 13.42(c) se ilustran los errores de alta y baja ganancia. En el caso de baja ganancia, todas las amplitudes de los escalones son menores que la ideal. En el caso de alta ganancia, todas las amplitudes de los escalones son mayores que la ideal. Esta situación puede deberse a un fallo de la resistencia de realimentación en el circuito del amplificador operacional.

Error de offset En la Figura 13.42(d) se ilustra el error de offset. Observe que, cuando la entrada binaria es 0000, la tensión de salida no es cero, y este valor de offset es el mismo para todos los escalones de la conversión. En esta situación, un fallo del amplificador operacional puede ser el culpable.

EJEMPLO 13.5

Se observa la salida del DAC de la Figura 13.43 cuando se aplica a las entradas una secuencia binaria ascendente de 4 bits. Identificar el tipo de error y sugerir un método para aislar el fallo.

Solución En este caso el DAC es no monotónico. El análisis de la salida revela que el dispositivo convierte la siguiente secuencia, en lugar de la secuencia binaria real aplicada a las entradas.

0010, 0011, 0010, 0011, 0110, 0111, 0110, 0111, 1010, 1011, 1010, 1011, 1110, 1111, 1110, 1111

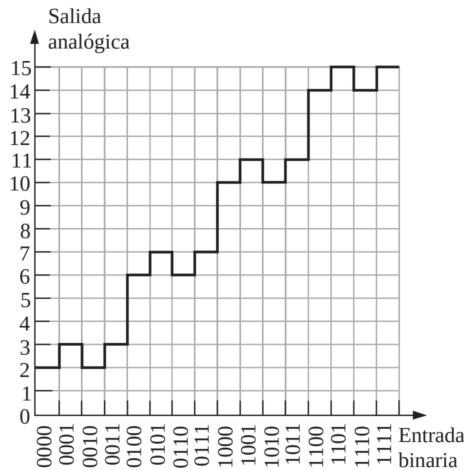


FIGURA 13.43

Aparentemente, el bit 2^1 se mantiene en el estado ALTO (1). Para localizar el problema, en primer lugar se comprueba el pin de la entrada del bit del dispositivo. Si cambia de estado, el fallo es interno, por lo que el DAC debería sustituirse. Si el pin externo no cambia de estado y está siempre a nivel ALTO, debe comprobarse si existe un cortocircuito externo a $+V$, que puede haber sido causado por un puente de soldadura en algún punto de la tarjeta del circuito.

Problema relacionado Determinar la salida de un DAC cuando se aplica a la entrada una secuencia binaria de 4 bits lineal y el bit 2^0 permanece a nivel ALTO.

El filtro de reconstrucción

La salida del DAC es una aproximación escalonada de la señal analógica original después de haber sido procesada por el DSP. El propósito del filtro de reconstrucción paso-bajo (en ocasiones denominado post-filtro) consiste en suavizar la salida del DAC eliminando las componentes de alta frecuencia generadas por las rápidas transiciones de la señal escalonada, como se ilustra en la Figura 13.44.

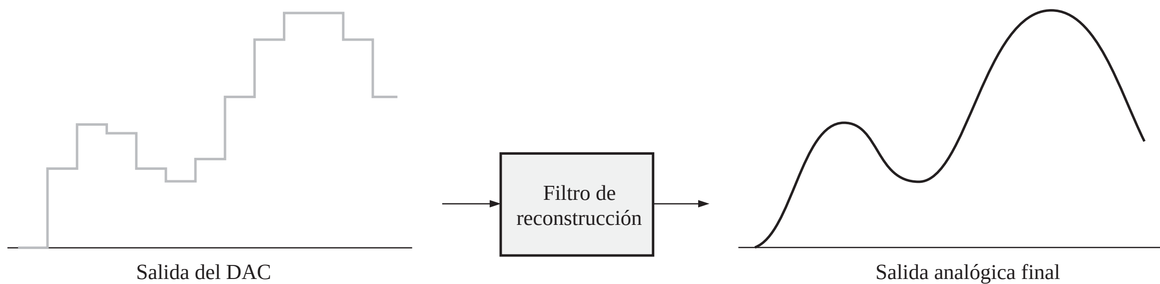


FIGURA 13.44 El filtro de reconstrucción suaviza la salida del DAC.

REVISIÓN DE LA SECCIÓN 13.5

1. ¿Qué desventaja tiene un DAC con ponderación binaria?
2. ¿Cuál es la resolución del DAC de 4 bits?
3. ¿Cómo se detecta el comportamiento no monotónico en un DAC?
4. ¿Qué efecto tiene en la salida del DAC una baja ganancia?

RESUMEN

- El procesamiento digital de la señal es el procesamiento digital de señales analógicas, habitualmente en tiempo real, con el propósito de modificar o realzar la señal de alguna manera.
- En general, un sistema de procesamiento digital de la señal está compuesto por un filtro *anti-aliasing*, un circuito de muestreo y retención, un convertidor analógico-digital, un DSP (procesador digital de señal), un convertidor digital-analógico y un filtro de reconstrucción.
- El muestreo convierte una señal analógica en una serie de impulsos, donde cada uno de ellos representa la amplitud de la señal en un determinado instante.
- El teorema de muestreo establece que la frecuencia de muestreo debe ser al menos dos veces superior a la frecuencia muestreada más alta (frecuencia de Nyquist).
- La conversión analógico-digital transforma una señal analógica en una serie de códigos digitales.
- Cuatro tipos de convertidores analógico-digitales (ADC) son el convertidor flash (paralelo), el de doble pendiente, el de aproximaciones sucesivas y el sigma-delta.
- Un DSP es un microprocesador especializado optimizado en términos de velocidad con el fin de procesar los datos a medida que se generan (tiempo real).
- La mayoría de los procesadores de tipo DSP están basados en una arquitectura Harvard, lo que quiere decir que existe una memoria de datos y una memoria de programa.
- Una operación *pipeline* está compuesta por las etapas de extracción, decodificación y ejecución.
- La conversión digital-analógica transforma una serie de códigos digitales que representan una señal analógica, con el fin de restaurar la señal analógica original.
- Dos tipos de convertidores digital-analógicos (DAC) son el convertidor de ponderación binaria y el convertidor en escalera *R/2R*.

PALABRAS CLAVE

Las palabras clave y otros términos que se han resaltado en negrita se encuentran en el glosario final del libro.

Aliasing El efecto creado cuando se muestrea una señal a menos de dos veces la frecuencia de señal máxima. El *aliasing* crea frecuencias no deseadas que interfieren con la frecuencia de la señal.

Convertidor analógico-digital (ADC) Un circuito utilizado para convertir una señal analógica a formato digital.

Convertidor digital-analógico (DAC) Un circuito utilizado para convertir en una señal analógica la representación digital de esa señal analógica.

Cuantificación El proceso mediante el que se asigna un código binario a cada valor muestreado durante la conversión analógico digital.

Decodificación Una etapa de las operaciones *pipeline* de un DSP en la que las instrucciones se asignan a unidades funcionales y se decodifican.

DSP *Digital Signal Processsor*, procesador digital de la señal; un tipo especial de microprocesador que procesa los datos en tiempo real.

Ejecución Una etapa de las operaciones *pipeline* de un DSP en la que se llevan a cabo las instrucciones decodificadas.

Extracción Una etapa de las operaciones *pipeline* de un DSP en la que se obtiene una instrucción de la memoria de programa.
Frecuencia de Nyquist La frecuencia más alta de señal que puede muestrearse a una determinada frecuencia de muestreo; se trata de una frecuencia igual o inferior a la mitad de la frecuencia de muestreo.
MFLOPS *Million Floating-Point Operations Per Second*, millones de operaciones en coma flotante por segundo.
MIPS *Million Instructions Per Second*, millones de instrucciones por segundo.
MMACS *Million Multiply/Accumulates Per Second*, millones de operaciones de multiplicación/acumulación por segundo.
Módulo DSP La unidad central de proceso de un DSP.
Muestreo El proceso de tomar un número suficiente de valores discretos en determinados puntos de una forma de onda, definiéndose con esos valores dicha forma de onda.
Pipeline Parte de la arquitectura de un DSP que permite procesar múltiples instrucciones.

AUTOTEST

Las respuestas se encuentran al final del capítulo.

1. Un ADC es un:
(a) *Alphanumeric Data Coder* (codificador de datos alfanuméricos)
(b) *Analog-to-digital Converter* (convertidor analógico-digital)
(c) *Analog Device Carrier* (portadora de dispositivo analógico)
(d) *Analog-to-Digital Comparator* (comparador analógico-digital)

2. Un DAC es un:
(a) *Digital-to-Analog Computer* (computadora digital-analógica)
(b) *Digital Analysis Calculator* (calculadora de análisis digital)
(c) *Data Accumulation Converter* (convertidor de acumulación de datos)
(d) *Digital-to-analog Converter* (convertidor digital-analógico)

3. Un sistema de procesamiento digital de la señal usualmente opera en:
(a) tiempo real **(b)** tiempo imaginario
(c) tiempo comprimido **(d)** tiempo de computadora

4. El muestreo de una señal analógica produce:
(a) una serie de impulsos que son proporcionales a la amplitud de la señal.
(b) una serie de impulsos que son proporcionales a la frecuencia de la señal.
(c) códigos digitales que representan la amplitud de la señal analógica.
(d) códigos digitales que representan el instante correspondiente a cada muestra.

5. De acuerdo con el teorema de muestreo, la frecuencia de muestreo debe ser:
(a) inferior a la mitad de la frecuencia más alta de la señal.
(b) mayor que dos veces la frecuencia más alta de la señal.
(c) inferior a la mitad de la frecuencia más baja de la señal.
(d) mayor que la frecuencia más baja de la señal.

6. El proceso de retención se produce:
(a) antes de cada muestreo.
(b) durante cada muestreo.
(c) después de la conversión analógico-digital.
(d) inmediatamente después de un muestreo.

7. El proceso de cuantificación:

(a) convierte la salida del bloque de muestreo y retención en un código binario.

(b) convierte un impulso de muestreo en un nivel.

(c) convierte una secuencia de códigos binarios en una señal analógica reconstruida.

(d) filtra las frecuencias no deseadas antes de que tenga lugar el muestreo.

8. Generalmente, una señal analógica puede reconstruirse de forma más precisa con:

(a) más niveles de cuantificación.

(b) menos niveles de cuantificación

(c) una mayor frecuencia de muestreo.

(d) una menor frecuencia de muestreo.

(e) las respuestas (a) y (c).

9. Un ADC flash utiliza**:**

(a) contadores **(b)** amplificadores operacionales

(c) un integrador **(d)** flip-flops

(e) las respuestas (a) y (c).

10. Un ADC de pendiente doble utiliza:

(a) contadores **(b)** amplificadores operacionales

(c) un integrador **(d)** un diferenciador

(e) las respuestas (a) y (c).

11. La salida de un ADC sigma-delta está compuesta por

(a) códigos binarios paralelos **(b)** datos de múltiples bits

(c) datos de un solo bit **(d)** una tensión diferencial

12. El término *arquitectura Harvard* implica:

(a) una CPU y una memoria principal.

(b) una CPU y dos memorias de datos.

(c) una CPU, una memoria de programa y una memoria de datos.

(d) una CPU y dos archivos de registros.

13. El número mínimo de registros de propósito general en los procesadores DSP de la serie TMS320C6000 es:

(a) 32 **(b)** 64 **(c)** 16 **(d)** 8

14. Las dos memorias internas en la serie TMS320C6000 tienen una capacidad de:

(a) 1 MB **(b)** 512 kB

(c) 64 kB **(d)** 21 kB

15. En las operaciones *pipeline* de la serie TMS320C6000, el número de instrucciones procesadas simultáneamente es:

(a) ocho **(b)** cuatro **(c)** dos **(d)** una

16. La etapa de las operaciones *pipeline* en la que las instrucciones se leen de la memoria se denomina:

(a) ejecución **(b)** acumulación

(c) decodificación **(d)** extracción

17. En un DAC de ponderación binaria, las resistencias de las entradas:

(a) determinan la amplitud de la señal analógica.

(b) determinan los pesos de las entradas digitales.

(c) limitan el consumo de potencia.

(d) evitan cargar a la fuente.

18. En un DAC *R/2R* hay:

(a) cuatro valores de resistencia.

(b) un valor de resistencia.

(c) dos valores de resistencia.

(d) un número de valores de resistencia igual al número de entradas.

PROBLEMAS

Las respuestas a los problemas impares se encuentran al final del libro.

SECCIÓN 13.1 Fundamentos del procesamiento digital de la señal

1. Explicar el propósito de la conversión analógica-digital.

2. Complete con los nombres funcionales apropiados el diagrama de bloques del sistema de procesamiento de la señal de la Figura 13.45

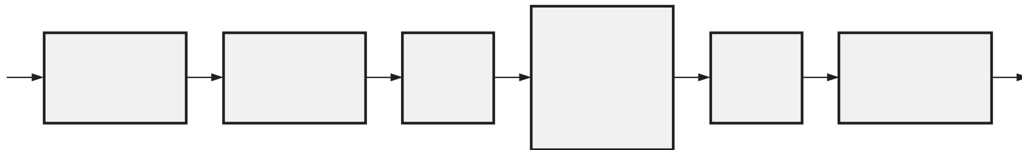


FIGURA 13.45

3. Explicar el propósito de la conversión digital-analógica.

SECCIÓN 13.2 Conversión de señales analógicas a formato digital

4. La forma de onda mostrada en la Figura 13.46 se aplica a un circuito de muestreo, muestreándose la señal cada 3 ms. Dibujar la salida del circuito de muestreo. Suponga que existe una correspondencia biunívoca de tensiones entre la entrada y la salida.

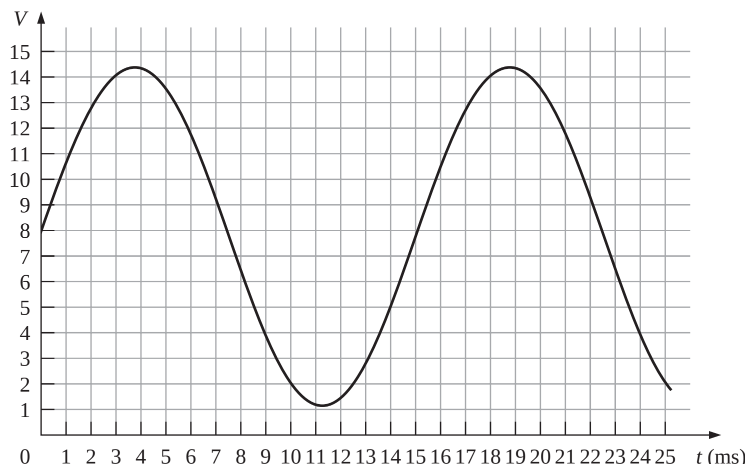


FIGURA 13.46

5. La salida del circuito de muestreo del Problema 4 se aplica a un circuito de retención. Obtener la salida del circuito retención.

6. Si la salida del circuito retención del Problema 5 se cuantiza utilizando dos bits, ¿cuál será la secuencia resultante de códigos binarios?

7. Repetir el Problema 6 utilizando una cuantificación de 4 bits.

8. **(a)** Reconstruir la señal analógica a partir de la cuantificación de 2 bits obtenida en el Problema 6.

   (b) Reconstruir la señal analógica a partir de la cuantificación de 4 bits obtenida en el Problema 7.

9. Dibuje la función analógica representada por la siguiente secuencia de números binarios:

   1111, 1110, 1101, 1100, 1010, 1001, 1000, 0111, 0110, 0101, 0100, 0101, 0110, 0111, 1000, 1001, 1010, 1011, 1100, 1100, 1100, 1011, 1010, 1001.

SECCIÓN 13.3 Métodos de conversión analógica-digital

10. La tensión de entrada a un determinado amplificador operacional inversor es de 10 mV y la salida es 2 V. ¿Cuál es la ganancia de tensión en bucle cerrado?

11. Para conseguir una ganancia de tensión en bucle cerrado de 330 con un amplificador inversor, ¿qué valor hay que utilizar para la resistencia de realimentación si R_i = 1,0 kΩ?

12. Determinar el código binario de salida de un ADC flash de 3 bits para la señal analógica de entrada de la Figura 13.47.

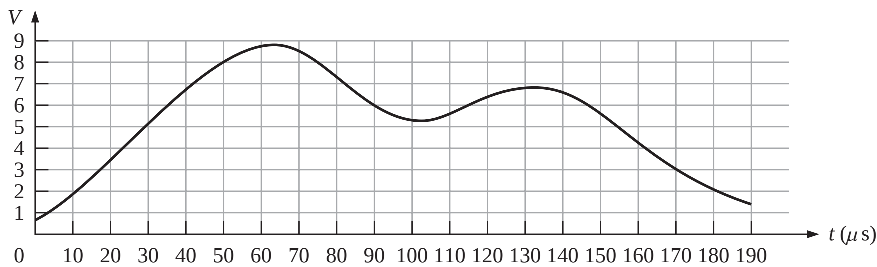


FIGURA 13.47

13. Repetir el Problema 12 para la forma de onda analógica de la Figura 13.48.

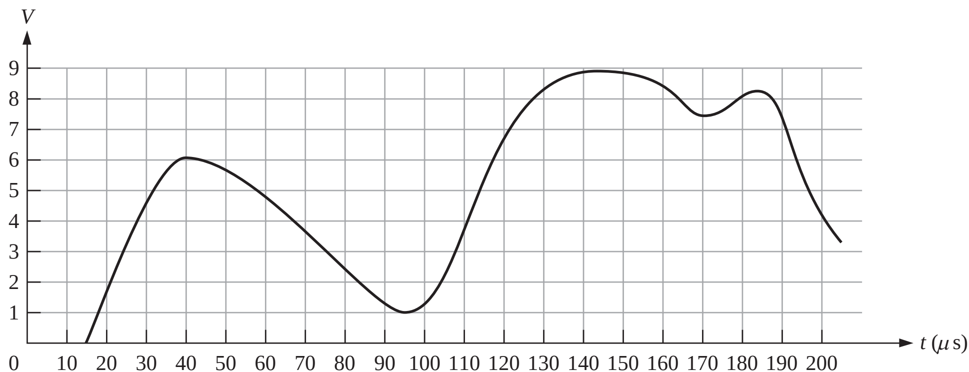


FIGURA 13.48

14. Para un cierto ADC de aproximaciones sucesivas de 2 bits, la salida máxima es de +8 V. Si se aplica una tensión constante de + 6V a la entrada analógica, determinar la secuencia de los estados binarios para el SAR.

15. Repetir el Problema 14 para un ADC de aproximaciones sucesivas de 4 bits.

16. Un ADC genera la siguiente secuencia de números binarios cuando se aplica una señal analógica a su entrada: 0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 0110, 0101, 0100, 0011, 0010, 0001, 0000.

(a) Reconstruir la entrada digitalmente.

(b) Si el ADC fallara de modo que el código 0111 estuviera ausente, ¿qué aspecto tendría la salida reconstruida?

SECCIÓN 13.4 Procesador digital de señal (DSP)

17. Un procesador DSP TMS320C062xx tiene instrucciones de 32 bits y opera a 2000 MIPS. ¿Cuántos bytes por segundo está procesando el DSP?

18. Si la frecuencia de reloj de un DSP TMS320C064xx es 400 MHz, ¿cuántas instrucciones puede proporcionar a la unidad funcional de la CPU en 1s?

19. ¿Cuántas operaciones en coma flotante puede realizar un DSP en 1 segundo si la especificación indica que es de 1000 MFLOPS?

20. Enumerar y describir las cuatro fases de la operación de extracción en un DSP de la serie TMS320C6000.

21. Enumerar y describir las dos fases de la operación de decodificación en un DSP de la serie TMS320C6000.

SECCIÓN 13.5 Métodos de conversión digital-analógica

22. En el DAC de 4 bits de la Figura 13.36, la resistencia de menor peso tiene un valor de 10 kΩ. ¿Qué valores deberían tener las otras resistencias de entrada?

23. Determinar la salida del DAC de la Figura 13.49(a) si se aplica a las entradas la secuencia de números de 4 bits indicada en la parte (b). Las entradas de datos tienen un valor de nivel BAJO de 0V y un valor de nivel ALTO de +5 V.

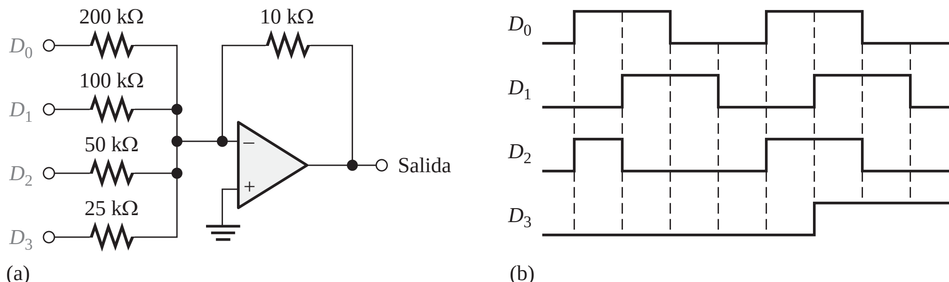


FIGURA 13.49

24. Repetir el Problema 23 para las entradas de la Figura 13.50

25. Determinar la resolución, expresada en porcentaje, para cada uno de los siguientes DAC:

(a) de 3 bits **(b)** de 10 bits **(c)** de 18 bits

26. Desarrollar un circuito para generar una secuencia de prueba binaria de 8 bits para la configuración de pruebas de la Figura 13.41.

27. Un DAC de 4 bits ha fallado de tal manera que el MSB está fijo en el estado 0. Dibujar la salida analógica cuando se aplica una secuencia binaria ascendente a las entradas.

28. Se aplica una secuencia binaria ascendente a un DAC de 4 bits y se observa la salida mostrada en la Figura 13.51. ¿Cuál es el problema?

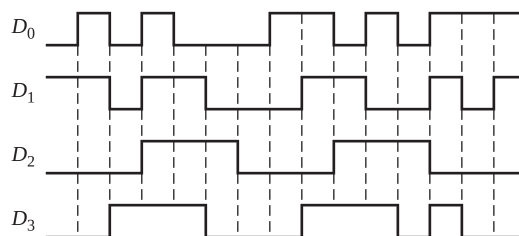


FIGURA 13.50

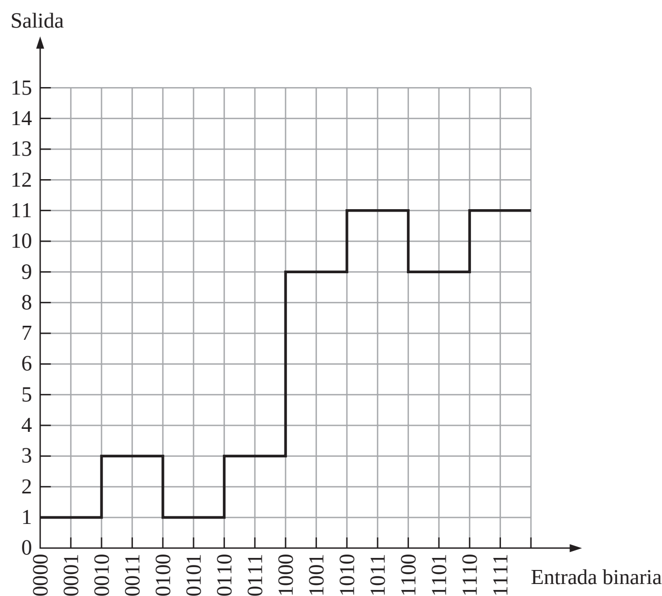


FIGURA 13.51

RESPUESTAS

REVISIONES DE LAS SECCIONES

SECCIÓN 13.1 **Fundamentos del procesamiento digital de la señal**

1. DSP quiere decir procesador digital de la señal.
2. ADC significa convertidor analógico-digital.
3. DAC significa convertidor digital-analógico.
4. El ADC transforma una señal analógica a un formato codificado binario.
5. El DAC transforma una señal con codificación binaria en forma analógica.

SECCIÓN 13.2 **Conversión de señales analógicas a formato digital**

1. Muestreo es el proceso de convertir una señal analógica en una serie de impulsos, donde cada uno de ellos representa la amplitud de la señal analógica.
2. Un valor muestreado se retiene para dar tiempo a convertirlo en un código binario.

3. La frecuencia mínima de muestreo es de 40 kHz.
4. La cuantificación es un proceso que convierte un nivel muestreado en un código binario.
5. El número de bits determina la precisión de la cuantificación.

SECCIÓN 13.3 Métodos de conversión analógica-digital

1. El método flash (paralelo) es el más rápido.
2. El método sigma-delta produce un flujo de datos de un solo bit.
3. Sí, el método de aproximaciones sucesivas tiene un tiempo de conversión fijo.
4. Los tipos de errores de salida de un ADC son los códigos que faltan, los códigos incorrectos y los errores de offset.

SECCIÓN 13.4 Procesador digital de la señal (DSP)

1. Una arquitectura Harvard tiene una CPU y dos memorias, una para los datos y otra para los programas.
2. El módulo DSP es la CPU.
3. Los DSP pueden ser de coma fija y de coma flotante.
4. Los dos tipos de memoria interna son la memoria de datos y la memoria de programa.
5. **(a)** MIPS (*Million Instructions Per Second*, millones de instrucciones por segundo)

   (b) MFLOPS (*Million Floating-Point Operations Per Second*, millones de operaciones de coma flotante por segundo)

   (c) MMACS (*Million Multiply/Acuumulates Per Second*, millones de operación de multiplicación/acumulación por segundo)
6. La técnica de *pipeline* permite procesar múltiples instrucciones simultáneamente.
7. Las etapas de una operación de *pipeline* son extracción, decodificación y ejecución.
8. Durante la extracción, las instrucciones se recuperan de la memoria de programa.

SECCIÓN 13.5 Métodos de conversión digital-analógica

1. En un DAC con ponderación binaria, cada resistencia tiene un valor diferente.
2. $(1/(2^4-1))100\% = 6{,}67\%$
3. En un DAC, una inversión de paso indica un comportamiento no monotónico.
4. Si la ganancia es baja, las amplitudes de los pasos en un DAC son inferiores a los valores ideales.

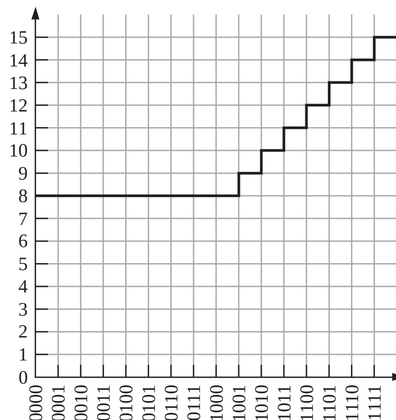


FIGURA 13.52

PROBLEMAS RELACIONADOS

13.1 100, 111, 100, 000, 011, 110. Sí, se pierde información.

13.2 *Véase* la Figura 13.52.

13.3 *Véase* la Figura 13.53.

13.4 (1/(216–1))100% = 0,00153%

13.5 *Véase* la Figura 13.54.

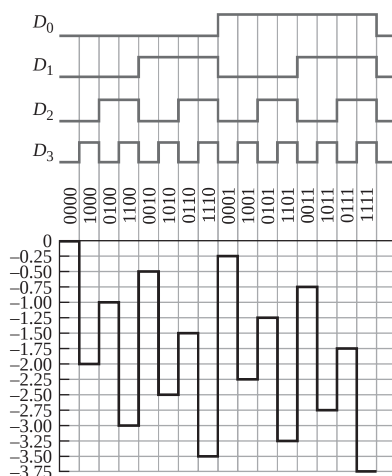


FIGURA 13.53

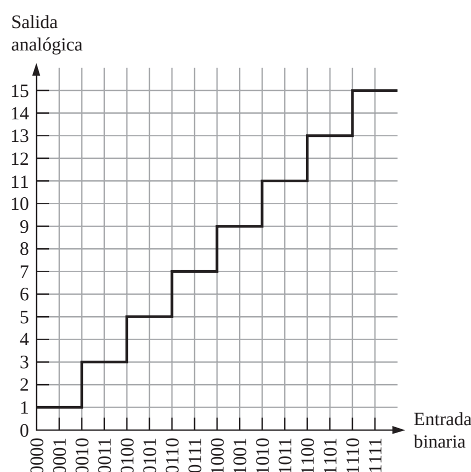


FIGURA 13.54

AUTOTEST

1. (b)	**2.** (d)	**3.** (a)	**4.** (a)	**5.** (b)	**6.** (d)	**7.** (a)	**8.** (e)	**9.** (b)
10. (e)	**11.** (c)	**12.** (c)	**13.** (a)	**14.** (c)	**15.** (a)	**16.** (d)	**17.** (b)	**18.** (c)

Referencias

Dahnoun, Naim. *Digital Signal Processing Implementation Using the TMS320C6000 DSP Platform*. Reading, Mass.: Addison-Wesley Longman. 2000.

Hayes, Monson. *Schaum's Outline of Digital Signal Processing*. Nueva York: McGraw-Hill. 1998.

Kuo, Sen, and Bob Lee. *Real-Time Digital Signal Processing: Implementations, Applications, and Experiments with the TMS320C55x*. Nueva York: John Wiley & Sons. 2001.

Lyons, Richard. *Understanding Digital Signal Processing*. Reading, Mass.: Addison-Wesley Longman. 1996.

Marven, Craig, and Gillian Ewers. *A Simple Approach to Digital Signal Processing*. Nueva York: John Wiley & Sons. 1996.

Oppenheim, Alan, and Ronald Schafer. *Digital Signal Processing*. Englewood Cliffs, N.J.: Prentice-Hall. 1974.

Orfanidis, Sophocles. *Introduction to Signal Processing*. Upper Saddle River, N.J.: Prentice-Hall. 1996.

Proakis, John, and Dimitris Manolakis. *Digital Signal Processing: Principles, Algorithms, and Applications,* 3ª ed. Upper Saddle River, N.J.: Prentice-Hall. 1996.

Steiglitz, Ken. *Digital Signal Processing Primer: With Applications to Digital Audio and Computer Music*. Reading, Mass.: Addison-Wesley Longman. 1996.

Williams, Douglas, and Vijay Madisetti. *Digital Signal Processing Handbook.* Boca Raton, Fl.: CRC Press. 1997.

14

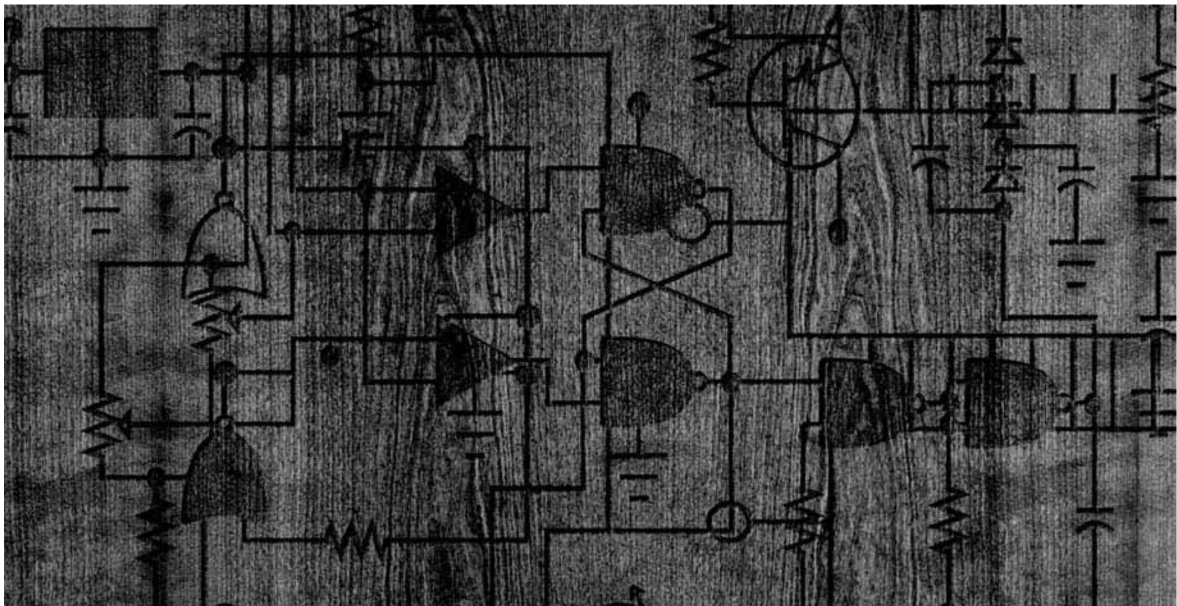
TECNOLOGÍAS DE LOS CIRCUITOS INTEGRADOS

CONTENIDO DEL CAPÍTULO

Antes de comenzar a leer este capítulo, debería estudiar la Sección 3.8.

OBJETIVOS DEL CAPÍTULO

- Determinar el margen de ruido de un dispositivo a partir de los parámetros de la hoja de características.
- Calcular la disipación de potencia de un dispositivo.
- Explicar cómo afecta el retardo de propagación a la frecuencia de funcionamiento o velocidad de un circuito.
- Interpretar el producto velocidad-potencia como una medida de rendimiento.
- Utilizar las hojas de características para obtener información sobre un dispositivo específico.



- Explicar qué es el *fan-out* de una puerta.
- Describir cómo funcionan las puertas TTL y CMOS básicas en el nivel de componentes.
- Establecer la diferencia entre salidas TTL *totem-pole* y salidas TTL en colector abierto, y entender las limitaciones y usos de cada una de ellas.
- Conectar circuitos en configuración AND-cableada.
- Describir el funcionamiento de los circuitos triestado.
- Terminar adecuadamente las entradas de las puertas no utilizadas.
- Comparar las características de las familias TTL y CMOS
- Manipular los dispositivos CMOS sin riesgos de daños debidos a las descargas electrostáticas.
- Indicar las ventajas de la tecnología ECL.
- Describir los circuitos PMOS y NMOS.
- Describir una celda E^2CMOS.

PALBRAS CLAVE

- TTL
- CMOS
- Inmunidad al ruido
- Margen de ruido
- Disipación de potencia
- Retardo de propagación
- *Fan-out*
- Fuente de corriente
- Sumidero de corriente
- Carga unidad
- Resistencia de *pull-up*
- Triestado
- *Totem pole*
- Colector abierto
- ECL
- E^2CMOS

INTRODUCCIÓN

La intención de este capítulo es que se utilice como capítulo “flotante”; es decir, todo el capítulo o partes del mismo pueden estudiarse en diferentes puntos del libro u omitirse por completo, dependiendo de los objetivos del curso. La Sección 3.8 debería estudiarse antes de comenzar con este capítulo.

En el Capítulo 3 (Sección 3.8) se aprendieron los fundamentos sobre las puertas lógicas en formato de CI. Este capítulo proporciona una introducción a la tecnología de circuitos utilizada para implementar dichas puertas, asi como otros tipos de dispositivos integrados.

Se abordan dos de las principales tecnologías, CMOS y TTL, y se definen sus parámetros de operación. Asimismo, se comparan las características operacionales de varias familias pertenecientes a dichas tecnologías de circuitos. También se presentan algunas otras tecnologías de circuitos. Es importante tener presente que la tecnología de circuitos concreta que se emplee para implementar una puerta lógica no tiene efecto sobre la operación lógica de la puerta. En términos de la operación de la tabla de verdad, una determinada puerta que se implemente con tecnología CMOS es igual que dicha puerta implementada con TTL. Las únicas diferencias de las puertas se encuentran en las características eléctricas, como por ejemplo la disipación de potencia, la velocidad de conmutación y la inmunidad al ruido.

14.1 PARÁMETROS Y CARACTERÍSTICAS DE OPERACIÓN BÁSICOS

Cuando trabaje con circuitos integrados digitales, no sólo debería familiarizarse con su funcionamiento lógico, sino también con sus propiedades de operación, como son los niveles de tensión, la inmunidad al ruido, la disipación de potencia, el *fan-out* y los retardos de propagación. En esta sección, se estudiarán los aspectos prácticos de estas propiedades.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Determinar las conexiones de alimentación y tierra. ■ Describir los niveles lógicos TTL y CMOS. ■ Explicar la inmunidad al ruido. ■ Determinar la disipación de potencia de un circuito lógico. ■ Definir los retardos de propagación de una puerta lógica. ■ Definir el producto velocidad-potencia y explicar su importancia. ■ Explicar la carga y el *fan-out* de los dispositivos TTL y CMOS.

Tensión de alimentación continua

El valor nominal de la tensión de alimentación continua (DC) para los dispositivos ***TTL*** (*Transistor-Transistor Logic*, lógica transistor-transistor) es de +5 V. TTL también se designa mediante T^2L. Los dispositivos ***CMOS*** (*Complementary Metal-Oxide Semiconductor*, metal óxido semiconductor complementario) están disponibles en diferentes categorías de tensiones de alimentación diferentes, +5V; +3,3 V; 2,5 V y 1,2 V. Aunque para simplificar se omite en los diagramas lógicos, esta tensión se conecta al pin V_{CC} de un circuito integrado, y la masa se conecta al pin de masa (GND). Tanto la alimentación como la masa se distribuyen internamente a todos los elementos del chip, como se ilustra en la Figura 14.1 para un encapsulado de 14 pines.

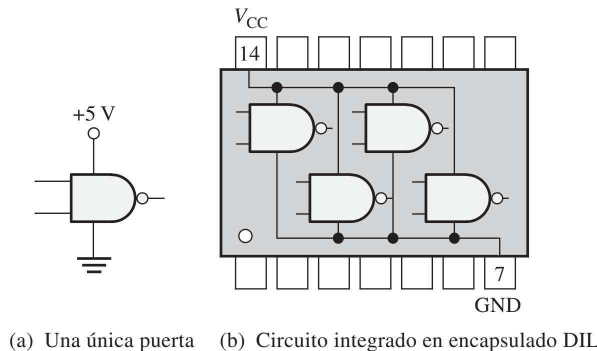


FIGURA 14.1 Ejemplo de las conexiones y distribución de V_{CC} y masa en un circuito integrado. Para simplificar, el resto de las conexiones se han omitido.

Niveles lógicos CMOS

En el Capítulo 1 se estudiaron brevemente los niveles lógicos. Existen cuatro especificaciones diferentes para los niveles lógicos: V_{IL}, V_{IH}, V_{OL} y V_{OH}. Para los circuitos CMOS, el rango de las tensiones de entrada (V_{IL}) que representan un nivel lógico BAJO (0 lógico) válido va de 0V a 1,5 V para la lógica de +5 V y de 0 V a 0,8 V para la lógica de 3,3 V. El rango de las tensiones de entrada (V_{IH}) que pueden representar un nivel ALTO (1 lógico) va de 3,5 a 5 V para la lógica de 5 V y de 2 V a 3,3V para la lógica de 3,3 V, como se indica en la Figura 14.2. El rango de valores entre 1,5 V y 3,5 V para la lógica de 5 V y el rango de 0,8 V a 2V para la lógica de 3,3 V son regiones de funcionamiento no predecible, y los valores comprendidos en dichos rangos

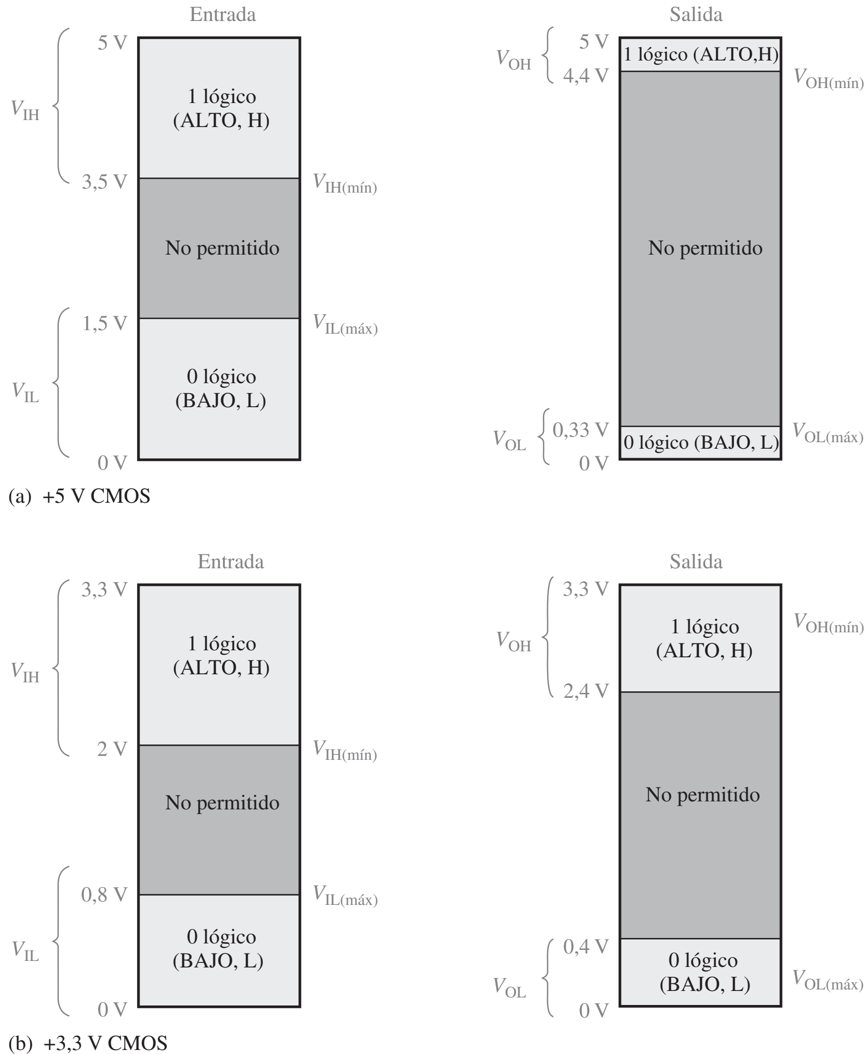


FIGURA 14.2 Niveles lógicos de entrada y de salida para los dispositivos CMOS.

no están permitidos. Cuando una tensión de entrada se encuentra en uno de estos rangos, puede ser interpretada como un nivel ALTO o un nivel BAJO por el circuito lógico. Por tanto, las puertas CMOS no pueden funcionar de forma fiable cuando las tensiones se encuentran dentro de uno de estos rangos no predecibles.

Los rangos de las tensiones de salida CMOS (V_{OL} y V_{OH}) para las lógicas de 5 V y 3,3 V se muestran en la Figura 14.2. Observe que la tensión de salida mínima a nivel ALTO, $V_{OH(mín)}$, es mayor que la tensión de entrada mínima a nivel ALTO, $V_{IH(mín)}$; y que la tensión de salida máxima a nivel BAJO, $V_{OL(máx)}$, es menor que la tensión de entrada máxima a nivel BAJO, $V_{IL(máx)}$.

Niveles lógicos TTL

En la Figura 14.3 se proporcionan los niveles lógicos de entrada y de salida para los dispositivos TTL. Al igual que para los dispositivos CMOS, existen cuatro especificaciones diferentes para los niveles lógicos: V_{IL}, V_{IH}, V_{OL} y V_{OH}.

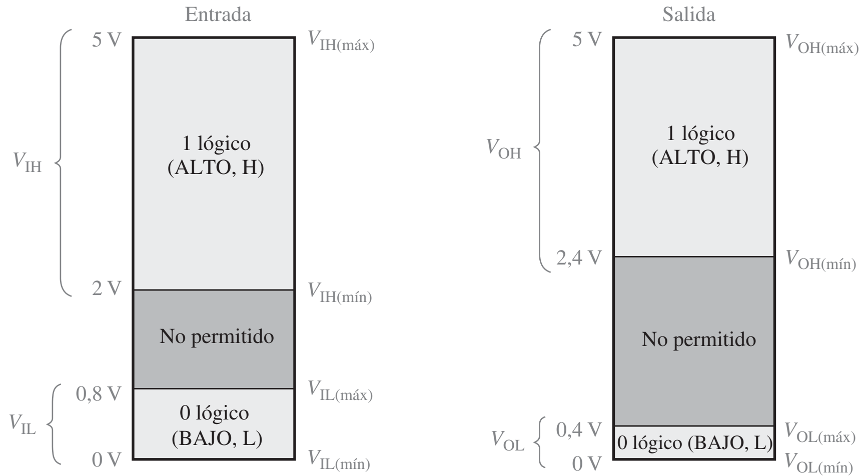


FIGURA 14.3 Niveles lógicos de entrada y salida para TTL.

Inmunidad al ruido

El ruido es una tensión no deseada que se induce en los circuitos eléctricos y que puede ser una amenaza para el correcto funcionamiento del circuito. Los cables y otros conductores internos del sistema pueden captar las radiaciones electromagnéticas de alta frecuencia de los conductores adyacentes, en los que las corrientes varían rápidamente, o de otras fuentes externas al sistema. También las fluctuaciones de tensión de la línea de alimentación son una forma de ruido de baja frecuencia.

Para no verse afectados adversamente por el ruido, los circuitos lógicos deben tener cierta ***inmunidad al ruido***, que es la capacidad de tolerar ciertas fluctuaciones de tensión no deseadas en sus entradas sin que cambie el estado de salida. Por ejemplo, si la tensión de ruido en la entrada de una puerta CMOS con lógica de +5 V hace que la tensión del nivel ALTO caiga por debajo de 3,5 V, el funcionamiento no será predecible, puesto que se encuentra en la región de operación no permitida (*véase* la Figura 14.2). Por tanto, la puerta puede interpretar la fluctuación por debajo de 3,5 V como un nivel BAJO, como se ilustra en la Figura 14.4(a). De forma similar, si el ruido hace que la entrada de una puerta pase por encima de 1,5 V en el nivel BAJO, se crea una condición indeterminada, como se ilustra en la parte (b).

Margen de ruido

La medida de la inmunidad al ruido de un circuito se denomina ***margen de ruido***, y se expresa en voltios. Para un determinado circuito, se especifican dos valores de margen de ruido: margen de ruido para el nivel ALTO (V_{NH}) y margen de ruido para el nivel BAJO (V_{NL}). Estos parámetros se definen mediante las siguientes ecuaciones:

Ecuación 14.1 $$V_{NH} = V_{OH(mín)} - V_{IH(mín)}$$

Ecuación 14.2 $$V_{NL} = V_{IL(máx)} - V_{OL(máx)}$$

En ocasiones, verá que el margen de ruido se expresa como un porcentaje de V_{CC}. A partir de la ecuación, puede ver que V_{NH} es la diferencia entre la salida a nivel ALTO menor posible de una puerta excitadora ($V_{OH(mín)}$) y la entrada a nivel ALTO menor posible que la puerta de carga puede tolerar ($V_{IH(mín)}$). El margen de

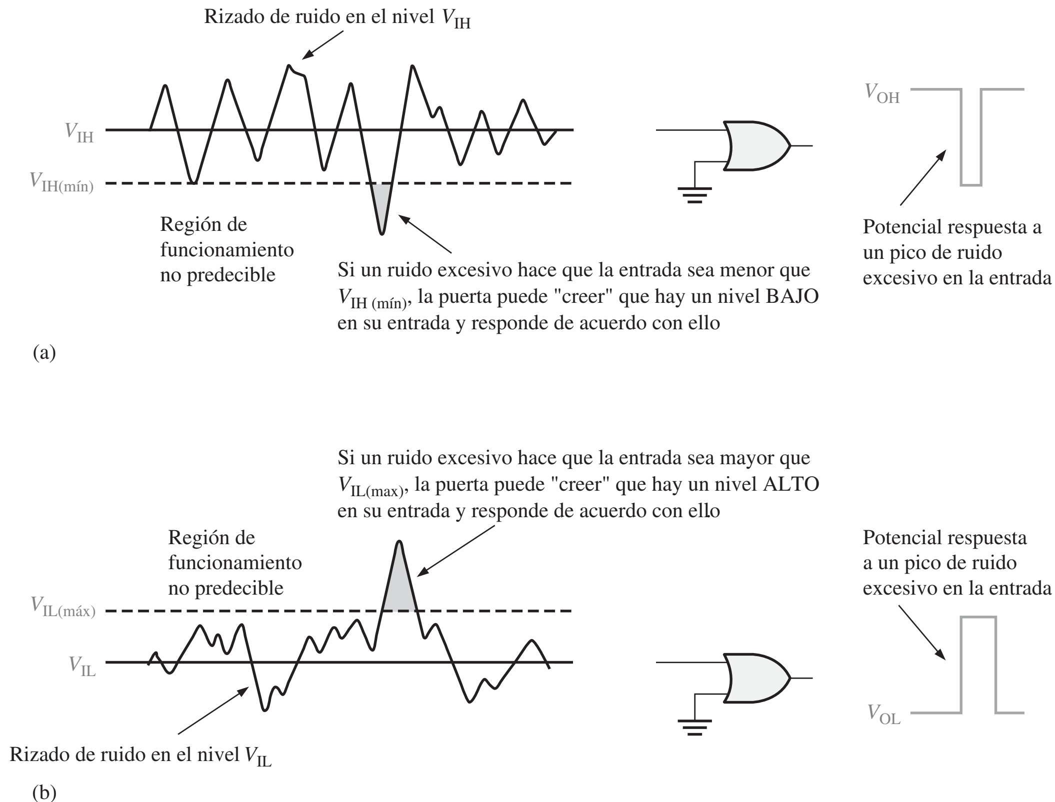


FIGURA 14.4 Efectos del ruido de entrada en el funcionamiento de la puerta.

ruido V_{NL} es la diferencia entre la entrada a nivel BAJO máxima posible que la puerta puede tolerar ($V_{IL(máx)}$) y la salida a nivel bajo máxima posible de la puerta excitadora ($V_{OL(máx)}$). Los márgenes de ruido se ilustran en la Figura 14.5.

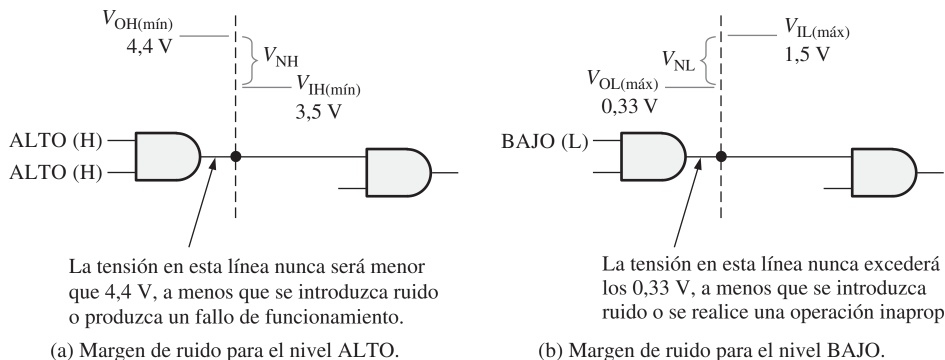


FIGURA 14.5 Márgenes de ruido. Los valores indicados son para dispositivos CMOS con lógica de 5 V, pero el principio se aplica a cualquier familia lógica.

EJEMPLO 14.1

Determinar los márgenes de ruido de los niveles ALTO y BAJO para TTL y CMOS, utilizando la información de las Figuras 14.2 y 14.3.

Solución Para CMOS de 5 V,

$$V_{IH(mín)} = 3,5\,V$$
$$V_{IL(máx)} = 1,5\,V$$
$$V_{OH(mín)} = 4,4\,V$$
$$V_{OL(máx)} = 0,33\,V$$
$$V_{NH} = V_{OH(mín)} - V_{IH(mín)} = 4,4\,V - 3,5\,V = \mathbf{0,9\,V}$$
$$V_{NL} = V_{IL(máx)} - V_{OL(máx)} = 1,5\,V - 0,33\,V = \mathbf{1,17\,V}$$

Para TTL

$$V_{IH(mín)} = 2\,V$$
$$V_{IL(máx)} = 0,8\,V$$
$$V_{OH(mín)} = 2,4\,V$$
$$V_{OL(máx)} = 0,4\,V$$
$$V_{NH} = V_{OH(mín)} - V_{IH(mín)} = 2,4\,V - 2\,V = \mathbf{0,4\,V}$$
$$V_{NL} = V_{IL(máx)} - V_{OL(máx)} = 0,8\,V - 0,4\,V = \mathbf{0,4\,V}$$

Una puerta TTL tiene una inmunidad de hasta 0,4 V de ruido para ambos estados de entrada, ALTO y BAJO.

Problema relacionado* Basándose en los cálculos anteriores de margen de ruido, ¿qué familia de dispositivos, CMOS de 5 V o TTL, debería utilizarse en entornos de alto ruido?

* Las respuestas se encuentran al final del capítulo.

Disipación de potencia

Como se indica en la Figura 14.6, por una puerta lógica circula corriente procedente de una fuente de alimentación continua. Cuando el estado de salida de la puerta es un nivel ALTO, circula la corriente I_{CCH}, y cuando el estado de salida es un nivel BAJO, circula la corriente I_{CCL}.

Veamos un ejemplo. Si se especifica una I_{CCH} de 1,5 mA cuando V_{CC} es 5 V, y si la puerta está en un estado de salida estático (no cambia) ALTO, la ***disipación de potencia*** (P_D) de la puerta es:

$$P_D = V_{CC} I_{CCH} = (5\,V)(1,5\,mA) = 7,5\,mW$$

Cuando se aplican impulsos a la puerta, su salida conmuta entre los estados ALTO y BAJO, por lo que la corriente de alimentación varía entre I_{CCH} e I_{CCL}. La disipación de potencia media depende del ciclo de trabajo y, usualmente, se especifica para un ciclo de trabajo del 50%. Cuando el ciclo de trabajo es el 50%, la salida está a nivel ALTO la mitad del tiempo, y la mitad restante está a nivel BAJO. Por tanto, la corriente de alimentación media es:

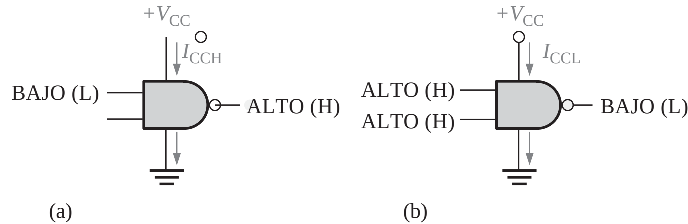


FIGURA 14.6 Corrientes de la fuente de alimentación DC. Se utiliza el convenio habitual para indicar la dirección de la corriente. El convenio para indicar el flujo de electrones es el contrario.

Ecuación 14.3 $$I_{CC} = \frac{I_{CCH} + I_{CCL}}{2}$$

La disipación de potencia media es

Ecuación 14.4 $$P_D = V_{CC} I_{CC}$$

EJEMPLO 14.2

Por una determinada puerta circulan 2 μA cuando su salida está a nivel ALTO, y 3,6 μA cuando está a nivel BAJO. ¿Cuál es la disipación de potencia media si V_{CC} es 5 V y la puerta funciona con un ciclo de trabajo del 50%?

Solución La corriente I_{CC} media es:

$$I_{CC} = \frac{I_{CCH} + I_{CCL}}{2} = \frac{2{,}0\,\mu A + 3{,}6\,\mu A}{2} = 2{,}8\,\mu A$$

La disipación de potencia media es:

$$P_D = V_{CC} I_{CC} = (5\,V)(2{,}8\,\mu A) = \mathbf{14\,\mu W}$$

Problema relacionado Una cierta puerta de un circuito integrado tiene una I_{CCH} = 1,5 μA e I_{CCL} = 2,8 μA. Determinar la disipación media para un ciclo de trabajo del 50% si V_{CC} es 5V.

La disipación de potencia en un circuito TTL es esencialmente constante dentro de su rango de frecuencias de operación. Sin embargo, la disipación de potencia en CMOS depende de la frecuencia. En condiciones de estática (DC) es extremadamente baja y aumenta cuando aumenta la frecuencia. En las curvas generales de la Figura 14.7 se muestran estas características. Por ejemplo, la disipación de potencia de una puerta TTL Schottky de bajo consumo es 2,2 mW siempre (constante). La disipación de potencia de una puerta HCMOS es 2,75 μW bajo condiciones estáticas y 170 μW a 100 kHz.

Retardo de propagación

Como ilustra la Figura 14.8, cuando una señal pasa (se propaga) a través de un circuito lógico, siempre experimenta un retardo temporal. Un cambio del nivel de salida siempre se produce un cierto tiempo, llamado **retardo de propagación**, después de que se ha realizado el cambio de nivel en la entrada.

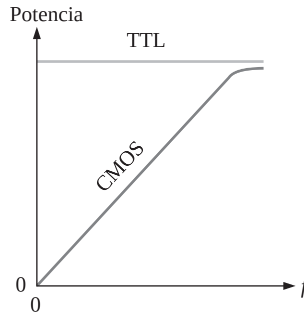


FIGURA 14.7 Curvas de potencia en función de la frecuencia para TTL y CMOS.

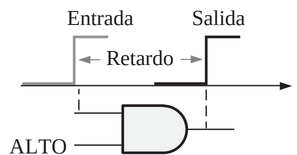


FIGURA 14.8 Ilustración básica del retardo de propagación.

Como se mencionó en el Capítulo 3, existen dos retardos de propagación específicos para las puertas lógicas:

- t_{PHL}: es el tiempo entre un determinado punto del impulso de entrada y el correspondiente punto del impulso de salida, cuando la salida cambia de nivel ALTO a nivel BAJO.
- t_{PLH}: es el tiempo entre un determinado punto del impulso de entrada y el correspondiente punto del impulso de salida, cuando la salida cambia de nivel BAJO a nivel ALTO.

En la Figura 14.9 se ilustran estos retardos de propagación, utilizando como referencias los puntos del 50% de los flancos de los impulsos.

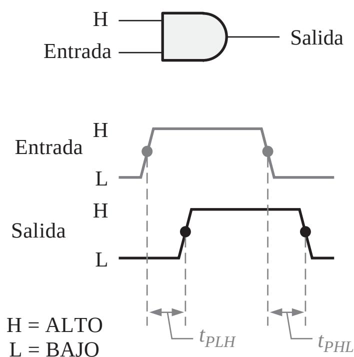


FIGURA 14.9 Retardos de propagación.

El retardo de propagación de una puerta limita la frecuencia a la que puede trabajar. Cuanto mayor es el retardo de propagación, menor es la frecuencia máxima. Luego un circuito de mayor velocidad tendrá un retardo de propagación menor. Por ejemplo, una puerta con un retardo de 3 ns es más rápida que otra que tenga un retardo de 10 ns.

Producto velocidad-potencia

Cuando, en una cierta aplicación, son importantes tanto el retardo de propagación como la disipación de potencia para la selección de la lógica que se debe utilizar, el producto velocidad-potencia es una base adecuada para la comparación de circuitos lógicos. Cuanto menor sea el producto velocidad-potencia, mejor. La unidad del producto velocidad-potencia es el picojulio (pJ). Por ejemplo, un HCMOS tiene un producto velocidad-potencia de 1,2 pJ a 100 kHz, mientras que un TTL de bajo consumo tiene un valor de 22 pJ.

Carga y *fan-out*

Cuando la salida de una puerta lógica se conecta a una o más entradas de otras puertas, como se muestra en la Figura 14.10, se genera una carga en la puerta excitadora. Existe un límite para el número de entradas de carga que una cierta puerta puede excitar. Este límite se denomina ***fan-out*** de la puerta.

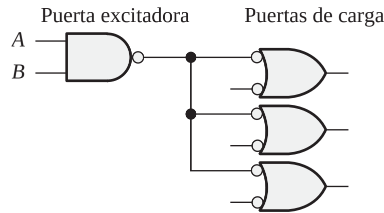


FIGURA 14.10 Carga de la salida de una puerta con las entradas de otras puertas.

Carga CMOS. La carga en CMOS difiere de la de TTL en que el tipo de transistores empleado en la lógica CMOS presenta una carga predominantemente capacitiva a la puerta excitadora, como ilustra la Figura 14.11. En este caso, las limitaciones vienen dadas por los tiempos de carga y descarga asociados con la resistencia de salida de la puerta excitadora y la capacitancia de entrada de las puertas de carga. Cuando la salida de la puerta excitadora está a nivel ALTO, la capacitancia de entrada de la puerta de carga se carga a través de la resistencia de salida de la puerta excitadora. Cuando la salida de la puerta excitadora está a nivel BAJO, la capacidad se descarga, como se indica en la Figura 14.11.

Cuando se añaden más entradas de puertas de carga a la salida de la puerta excitadora, la capacitancia total aumenta, puesto que las capacitancias de entrada están en paralelo. Este aumento de la capacitancia aumenta los tiempos de carga y de descarga, por lo que se reduce la frecuencia máxima a la que la puerta puede funcionar. Por tanto, el *fan-out* de una puerta CMOS depende de la frecuencia de operación. Cuantas menos entradas de carga haya, mayor será la frecuencia máxima.

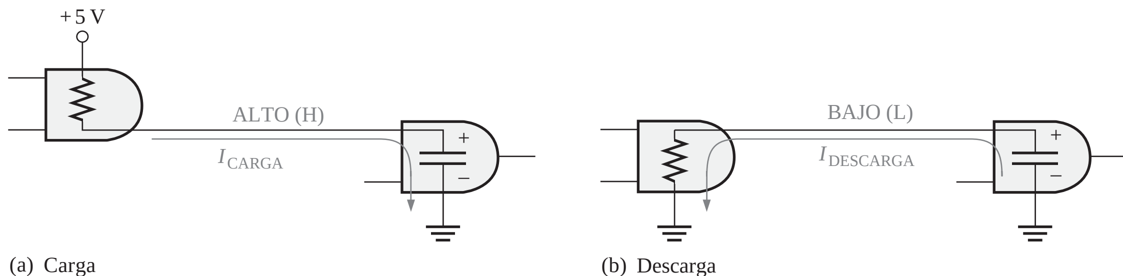


FIGURA 14.11 Carga capacitiva de una puerta CMOS.

Carga TTL. Una puerta excitadora TTL entrega (fuente) corriente a las entradas de las puertas de carga en el estado ALTO (I_{IH}) y absorbe corriente (sumidero) de las puertas de carga en el estado BAJO (I_{IL}). En la Figura 14.12 se ilustran de forma simplificada el funcionamiento como ***fuente de corriente*** y como ***sumidero de corriente***, donde las resistencias representan las resistencias internas de entrada y salida de la puerta para ambas condiciones.

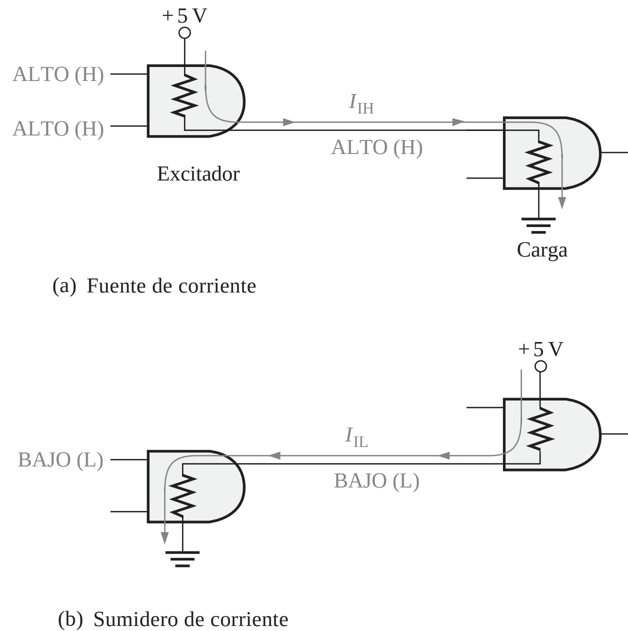


FIGURA 14.12 Fuente y sumidero de corriente en las puertas lógicas.

Cuantas más puertas de carga se conectan a la puerta excitadora, mayor es la carga de la misma. La corriente de fuente total aumenta con cada puerta de carga que se añade, como se muestra en la Figura 14.13. Al aumentar esta corriente, la caída de tensión interna de la puerta excitadora aumenta, haciendo que la tensión de salida V_{OH} disminuya. Si se conecta un número excesivo de puertas de carga, la tensión V_{OH} cae por debajo de su valor mínimo, $V_{OH(mín)}$, reduciéndose el margen de ruido del nivel ALTO, lo que compromete el funcionamiento del circuito. También ocurre que, cuando la corriente de fuente aumenta, la disipación de potencia de la puerta excitadora aumenta.

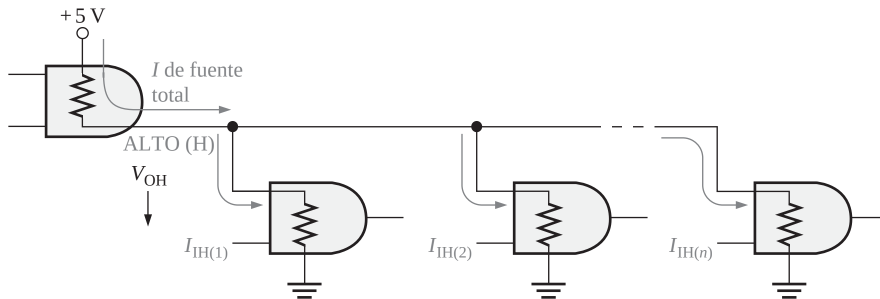


FIGURA 14.13 Carga TTL para el nivel ALTO.

El *fan-out* es el número máximo de entradas de carga que se pueden conectar sin afectar adversamente a las características de operación especificadas de la puerta. Por ejemplo, las TTL Schottky de bajo consumo (LS) tienen un *fan-out* de 20 unidades de carga. Una entrada de la misma familia lógica que la puerta excitadora se *llama* ***unidad de carga***.

La corriente total de sumidero también aumenta con cada entrada de carga que se añade, como muestra la Figura 14.14. Al aumentar esta corriente, la caída de tensión interna de la puerta excitadora aumenta, haciendo que V_{OL} aumente. Si se añade un número demasiado grande de cargas, V_{OL} se hará mayor que $V_{OL(máx)}$, reduciéndose el margen de ruido para el nivel BAJO.

En TTL, la capacidad de la corriente de sumidero (estado de salida a nivel BAJO) es el factor más crítico en la determinación del *fan-out*.

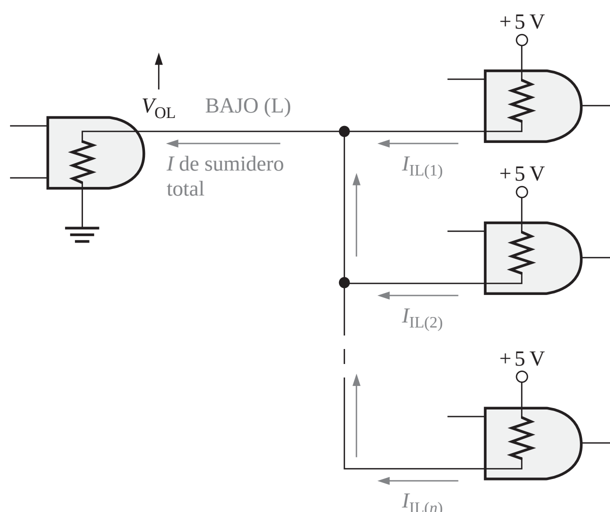


FIGURA 14.14 Carga TTL en estado BAJO.

REVISIÓN DE LA SECCIÓN 14.1

Las respuestas se encuentran al final del capítulo

1. Definir V_{IH}, V_{IL}, V_{OH} y V_{OL}.
2. ¿Qué es mejor, disponer de un valor de margen de ruido alto o bajo?
3. La puerta *A* tiene un retardo de propagación mayor que la puerta *B*. ¿Qué puerta puede trabajar a una frecuencia mayor?
4. ¿Cómo afecta una carga excesiva al margen de ruido de una puerta?

14.2 CIRCUITOS CMOS

En esta sección se presenta la circuitería interna CMOS básica y su funcionamiento. Las siglas CMOS corresponden a *Complementary Metal-Oxide Semiconductor* (metal-óxido semiconductor complementario). El término *complementario* se refiere a la utilización de dos tipos de transistores en el circuito de salida. Se usan MOSFET (MOS *Field-Effect Transistor*, transistor de efecto de campo MOS) de canal-*n* y de canal-*p*.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Identificar un MOSFET por su símbolo. ■ Explicar la acción de conmutación de un MOSFET. ■ Describir el funcionamiento básico de un circuito inversor CMOS. ■ Describir el funcionamiento básico de las puertas NAND y NOR CMOS. ■ Explicar el funcionamiento de una puerta CMOS con salida en drenador abierto. ■ Explicar el funcionamiento de las puertas CMOS de tres estados. ■ Enumerar las precauciones requeridas cuando se trabaja con dispositivos CMOS.

El MOSFET

Los transistores de efecto de campo de semiconductor de metal-óxido (**MOSFET**) son los elementos activos de conmutación de los circuitos CMOS. Estos dispositivos difieren enormemente, tanto en la construcción como en el funcionamiento interno de los transistores bipolares utilizados en los circuitos TTL pero, básicamente, su acción de conmutación es la misma. Idealmente, funcionan como interruptores abiertos o cerrados, dependiendo de la entrada.

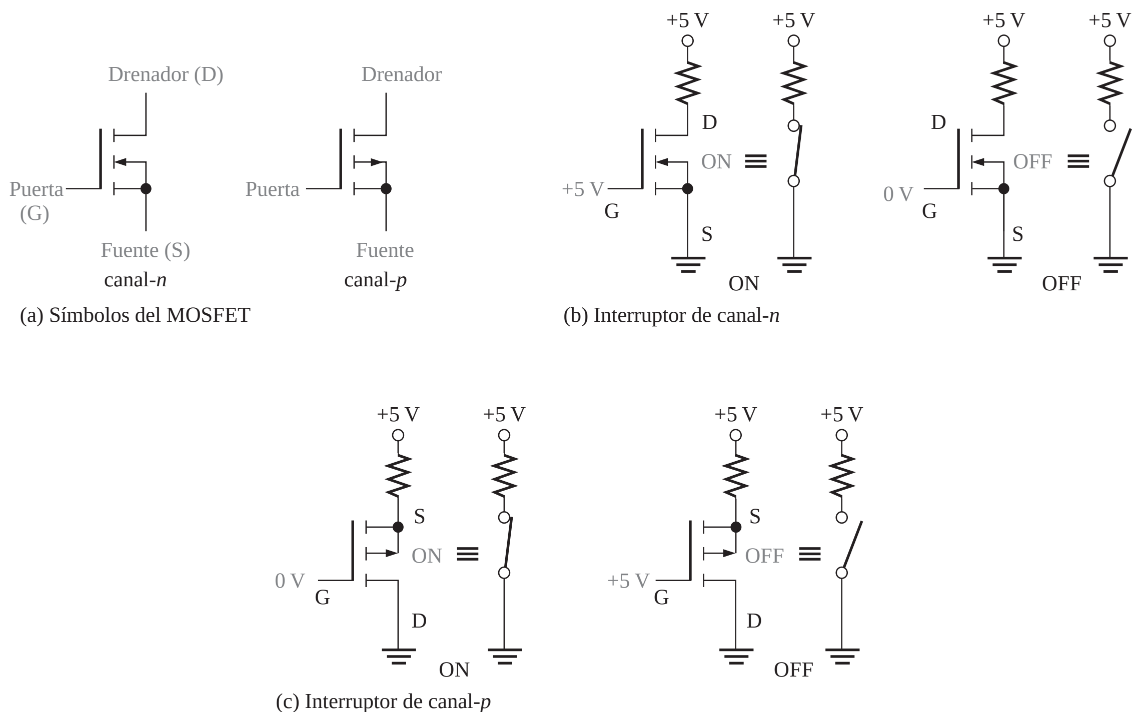


FIGURA 14.15 Símbolos básicos y acción de conmutación de los MOSFET.

La Figura 14.15(a) presenta los símbolos para los MOSFET de canal-*n* y de canal-*p*. Como se puede ver, los tres terminales de un MOSFET son: **puerta**, **drenador** y **fuente**. Cuando la tensión de puerta de un MOSFET de canal-*n* es más positiva que la de fuente, el MOSFET conduce (*saturación*) e, idealmente, se comporta como un conmutador cerrado entre el drenador y la fuente. Cuando la tensión puerta-fuente es cero, el MOSFET no conduce (está al *corte*) e, idealmente, se comporta como un interruptor abierto entre el drenador y la fuente. Este modo de operación se ilustra en la Figura 14.15(b). Los MOSFET de canal-*p* funcionan con polaridades de tensión opuestas, como se muestra en la parte (c).

Algunas veces, se utiliza un símbolo simplificado para el MOSFET, como el mostrado en la Figura 14.16.

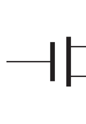


FIGURA 14.16 Símbolo simplificado del MOSFET.

Inversor CMOS

La lógica MOS complementaria (CMOS) utiliza pares complementarios MOSFET como elemento básico. Un par complementario emplea transistores MOSFET de canal-*p* y de canal-*n*, como muestra el circuito inversor de la Figura 14.17.

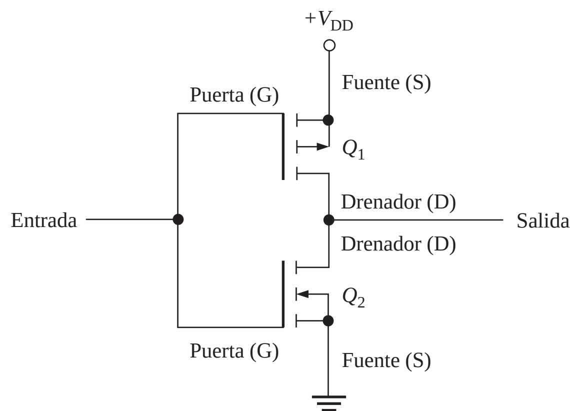
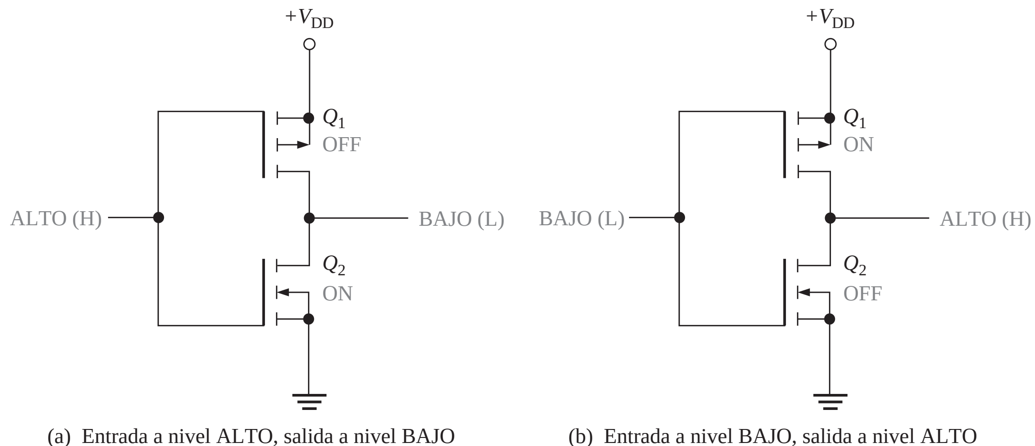


FIGURA 14.17 Circuito inversor CMOS.

Cuando se aplica un nivel ALTO a la entrada, el MOSFET de canal-*p* Q_1 no conduce, y el MOSFET de canal-*n* Q_2 conduce (se satura), como se indica en la Figura 14.18(a). Esta condición hace que la salida se conecte a tierra a través de la resistencia de *conducción (on)* de Q_2, produciendo un nivel BAJO de salida.



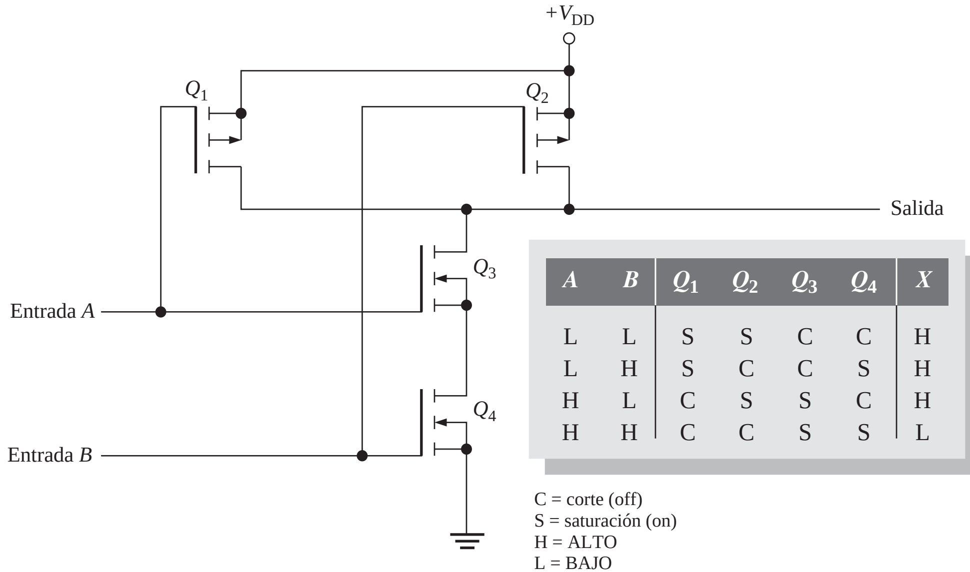
(a) Entrada a nivel ALTO, salida a nivel BAJO
(b) Entrada a nivel BAJO, salida a nivel ALTO

FIGURA 14.18 Funcionamiento de un inversor CMOS.

Cuando se aplica un nivel BAJO a la entrada, Q_1 se satura y Q_2 no, como se muestra en la Figura 14.18(b). Esta condición hace que la salida se conecte a $+V_{DD}$ (tensión de alimentación continua) a través de la resistencia de conducción de Q_1, produciendo un nivel ALTO de salida.

Puerta NAND CMOS

La Figura 14.19 muestra una puerta NAND CMOS con dos entradas. Observe la disposición de los pares complementarios (dispositivos MOSFET de canal-*n* y canal-*p*).



A	*B*	Q_1	Q_2	Q_3	Q_4	*X*
L	L	S	S	C	C	H
L	H	S	C	C	S	H
H	L	C	S	S	C	H
H	H	C	C	S	S	L

FIGURA 14.19 Circuito de una puerta CMOS NAND.

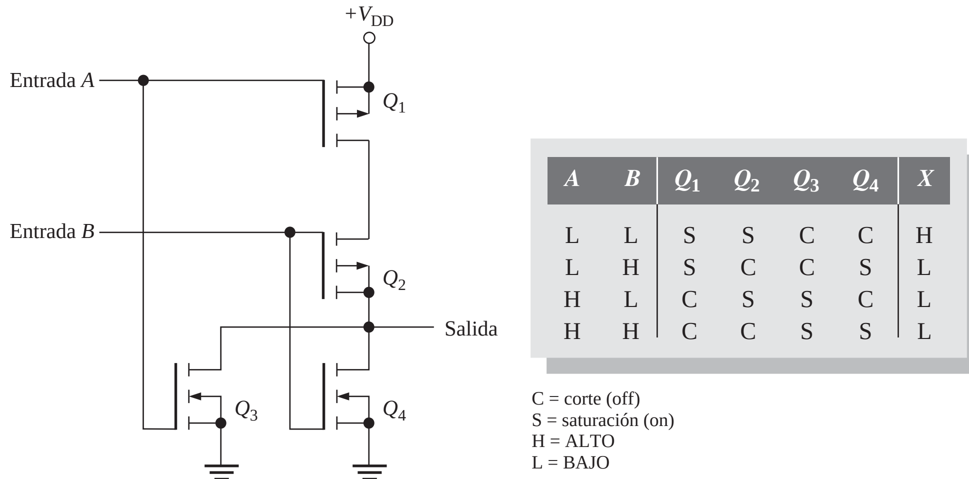
El funcionamiento de una puerta NAND CMOS es el siguiente:

- Cuando ambas entradas están a nivel BAJO, Q_1 y Q_2 se saturan y, Q_3 y Q_4 no. La salida se fuerza al nivel ALTO a través de las resistencias de conducción de Q_1 y Q_2, conectadas en paralelo.
- Cuando la entrada *A* está a nivel BAJO y la entrada *B* esta a nivel ALTO, Q_1 y Q_4 se saturan, mientras que Q_2 y Q_3 no. La salida se fuerza a nivel ALTO a través de la baja resistencia de conducción de Q_1.
- Cuando la entrada *A* está a nivel ALTO y la entrada *B* está a nivel BAJO, Q_1 y Q_4 no conducen, mientras que Q_2 y Q_3 se saturan. La salida se fuerza a nivel ALTO a través de la baja resistencia de conducción de Q_2.
- Por último, cuando ambas entradas están a nivel ALTO, Q_1 y Q_2 no conducen, y Q_3 y Q_4 se saturan. En este caso, la salida se fuerza al nivel BAJO a través de las resistencias de conducción de Q_3 y $Q_{4,}$ en serie con tierra.

Puerta NOR CMOS

La Figura 14.20 muestra una puerta NOR CMOS con dos entradas. Observe la disposición de los pares complementarios.

El funcionamiento de una puerta NOR CMOS es el siguiente:



A	B	Q_1	Q_2	Q_3	Q_4	X
L	L	S	S	C	C	H
L	H	S	C	C	S	L
H	L	C	S	S	C	L
H	H	C	C	S	S	L

C = corte (off)
S = saturación (on)
H = ALTO
L = BAJO

FIGURA 14.20 Circuito de una puerta CMOS NOR.

- Cuando ambas entradas están a nivel BAJO, Q_1 y Q_2 se saturan, y Q_3 y Q_4 no. La salida se fuerza al nivel ALTO a través de las resistencias de conducción de Q_1 y Q_2, situadas en serie.
- Cuando la entrada *A* está a nivel BAJO y la entrada *B* esta a nivel ALTO, Q_1 y Q_4 se saturan, mientras que Q_2 y Q_3 no. La salida se fuerza a nivel BAJO a través de la resistencia de conducción de Q_4, conectada a tierra.
- Cuando la entrada *A* está a nivel ALTO y la entrada *B* esta a nivel BAJO, Q_1 y Q_4 no conducen y Q_2 y Q_3 se saturan. La salida se fuerza a nivel bajo a través de la resistencia de conducción de Q_3, conectada a tierra.
- Cuando ambas entradas están a nivel ALTO, Q_1 y Q_2 no conducen, mientras que Q_3 y Q_4 se saturan. La salida se fuerza al nivel BAJO a través de las resistencias de conducción de Q_3 y Q_4, en paralelo y conectadas a tierra.

Puertas en drenador abierto

El término *drenador abierto* quiere decir que el drenador del transistor de salida no está conectado y debe conectarse externamente a V_{DD} a través de una carga. Las puertas en drenador abierto son la contrapartida CMOS de las puertas TTL en colector abierto (estudiado en la Sección 14.3). Un circuito de salida en drenador abierto es un único MOSFET de canal-*n*, como muestra la Figura 14.21(a). Debe utilizarse una ***resistencia de pull-up*** externa, como se indica en la parte (b), para producir un estado de salida ALTO. Las salidas en drenador abierto también pueden conectarse en configuración AND-cableada, concepto que se explica en la sección siguiente, dedicada a las puertas TTL.

Puertas CMOS triestado

Las salidas triestado están disponibles tanto en la lógica CMOS como en la TTL. La salida ***triestado*** combina las ventajas de los circuitos *totem-pole* y en colector abierto. Como recordará, los tres estados de salida son ALTO, BAJO y alta impedancia (**Alta-Z**). Cuando se selecciona el modo normal de operación de nivel lógico, lo cual depende del estado de la entrada de habilitación, un circuito triestado funciona del mismo modo que una puerta normal. Cuando se selecciona el modo de operación de alta impedancia, la salida se desconec-

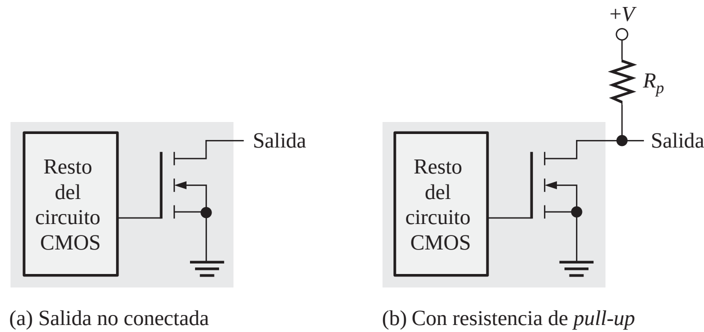


FIGURA 14.21 Puertas CMOS en drenador abierto.

ta de forma efectiva del resto del circuito mediante la circuitería interna. La Figura 14.22 ilustra el funcionamiento de un circuito triestado. El triángulo invertido (∇) indica una salida triestado.

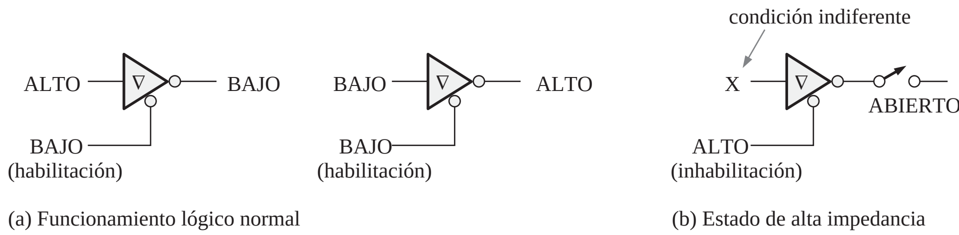


FIGURA 14.22 Los tres estados de un circuito triestado.

La circuitería de una puerta CMOS triestado, como muestra la Figura 14.23, permite poner al corte a ambos transistores de salida, Q_1 y Q_2, a un mismo tiempo, desconectando la salida del resto del circuito.

Cuando la entrada de habilitación está a nivel BAJO, el dispositivo se activa para funcionamiento normal. Cuando la entrada de habilitación está a nivel ALTO, Q_1 y Q_2 se bloquean y el circuito está en estado de alta impedancia.

Precauciones para la manipulación de CMOS

Como ya hemos visto, todos los dispositivos CMOS son susceptibles de ser dañados por las descargas electrostáticas. Por tanto, debe trabajarse con ellos poniendo un especial cuidado. Tenga en cuenta las siguientes precauciones:

1. Todos los dispositivos CMOS se suministran embalados en espuma conductora para evitar la formación de cargas electrostáticas. Cuando se retiran de la espuma, los pines no deberían tocarse.
2. Cuando se retiran del material protector, los dispositivos deben colocarse con los pines hacia abajo sobre una superficie a masa, tal como una placa metálica. No coloque los dispositivos CMOS sobre espumas de poliestireno o bandejas de plástico.
3. Todas las herramientas, equipos de pruebas y bancos de trabajo metálicos deberían estar conectados a tierra. En determinados entornos, la persona que trabaja con dispositivos CMOS debería conectar su muñeca a masa por medio de un cable y una resistencia serie de alto valor. Esta resistencia evita sufrir una descarga si la persona entra en contacto con una fuente de tensión.

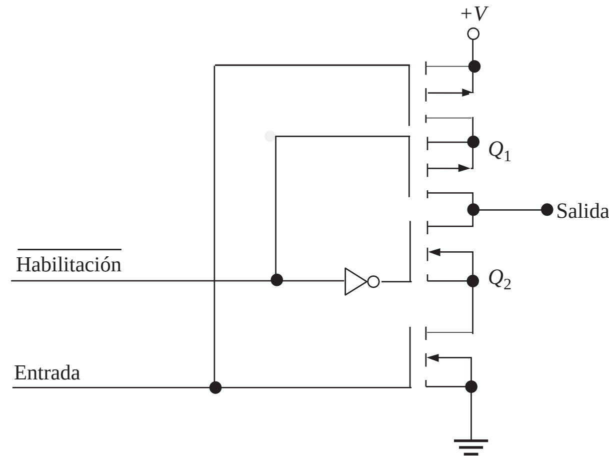


FIGURA 14.23 Inversor CMOS de tres estados.

4. No inserte los dispositivos CMOS (o cualquier otro dispositivo) en los zócalos de una tarjeta de circuito impreso con la alimentación conectada.
5. Todas las entradas no utilizadas deberían conectarse a la tensión de alimentación o a tierra, como se indica en la Figura 14.24. Si se deja en circuito abierto, una entrada puede adquirir carga electrostática y "flotar" en niveles impredecibles.

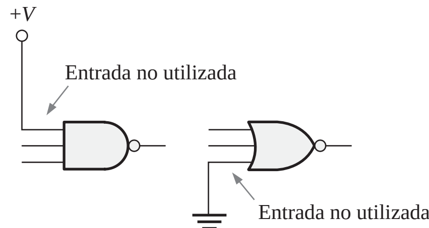


FIGURA 14.24 Tratamiento de las entradas CMOS no utilizadas.

6. Después del ensamblado de las tarjetas de circuito de impreso, éstas deben protegerse durante el almacenamiento o transporte insertando sus conectores en espuma conductora. Los pines de entrada y salida CMOS se pueden proteger también con resistencias de alto valor conectadas a masa.

REVISIÓN DE LA SECCIÓN 14.2

1. ¿Qué tipo de transistor se usa en la lógica CMOS?
2. ¿Qué significa el término *MOS complementario*?
3. ¿Por qué los dispositivos CMOS deben manipularse con cuidado?

14.3 CIRCUITOS TTL

En esta sección se cubre el funcionamiento interno de los circuitos lógicos TTL con salidas *totem-pole*. También se tratará el funcionamiento de las puertas TTL con salida en colector abierto, y el de las puertas triestado.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Identificar mediante su símbolo al transistor de unión bipolar (BJT, *Bipolar Junction Transistor*). ■ Describir la acción de conmutación de un BJT. ■ Describir el funcionamiento básico de un circuito inversor TTL. ■ Describir el funcionamiento básico de la puerta TTL NAND. ■ Explicar qué es una salida *totem-pole*. ■ Explicar el funcionamiento y utilización de una puerta TTL con salida en colector abierto. ■ Explicar el funcionamiento de una puerta con salida triestado.

El transistor de unión bipolar

El transistor de unión **bipolar** (**BJT**) es el elemento activo de conmutación utilizado en todos los circuitos TTL. La Figura 14.25 muestra el símbolo de un BJT *npn* con sus tres terminales: **base, emisor** y **colector**. Un BJT tiene dos **uniones**: la unión base-emisor y la unión base-colector.

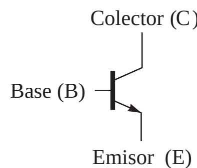
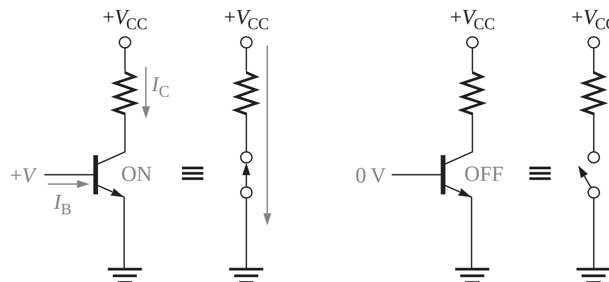


FIGURA 14.25 Símbolo del BJT.

La operación básica de conmutación es la siguiente: cuando la base está aproximadamente a 0,7 V por encima del emisor y se proporciona suficiente corriente a la base, el **transistor** conduce y entra en saturación. Idealmente, en saturación, el transistor actúa como un interruptor cerrado entre el colector y el emisor, como ilustra la Figura 14.26(a). Cuando la base está a menos de 0,7 V por encima del emisor, el transistor no conduce y actúa como un interruptor abierto entre el colector y el emisor, como muestra la parte (b). Resumiendo, un nivel ALTO en la base pone en conducción y satura (*on*) al transistor, por lo que actúa como un interruptor cerrado. Un nivel BAJO en la base bloquea (*off*) al transistor, el cual se comporta como un interruptor abierto. En la lógica TTL, existen algunos BJT con múltiples emisores.



(a) Transistor saturado (ON) e interruptor equivalente ideal. (b) Transistor bloqueado (OFF) e interruptor equivalente ideal

FIGURA 14.26 Conmutación ideal del BJT. Se utiliza el convenio habitual para indicar la dirección de la corriente. El convenio para indicar el flujo de electrones es el contrario.

Inversor TTL

La función lógica de un inversor o de cualquier tipo de puerta es siempre la misma, independientemente del tipo de tecnología de circuitos que se utilice. La Figura 14.27 muestra un circuito TTL estándar para un inver-

sor. En esta figura, Q_1 es el transistor de acoplamiento de entrada y D_1 es el **diodo** de fijación del nivel de entrada. El transistor Q_2 es el *divisor de fase* y la combinación de Q_3 y Q_4 forma el circuito de salida, a menudo denominado disposición ***totem-pole***.

Cuando la entrada es un nivel ALTO, la unión base-emisor de Q_1 se **polariza en inversa**, y la unión base-colector se **polariza en directa**. Esta condición permite que la corriente atraviese R_1 y la unión base-colector de Q_1, llevando a Q_2 a la saturación. Como resultado, Q_2 excita a Q_3, y su tensión de colector, que es la de salida, es próxima al potencial de tierra. Por tanto, se obtiene una salida a nivel BAJO para una entrada a nivel ALTO. También, en este mismo instante, el colector de Q_2 está a un nivel de tensión suficientemente bajo como para mantener bloqueado a Q_4.

Cuando la entrada está a nivel BAJO, la unión base-emisor de Q_1 se polariza en directa y la unión base-colector se polariza en inversa, por lo que se genera una corriente a través de R_1 y de la unión base-emisor de Q_1. Un nivel BAJO proporciona un camino a tierra para la corriente. En la base de Q_2 no hay corriente, por lo que no conduce. El colector de Q_2 está a nivel ALTO, lo que pone en conducción a Q_4. El transistor Q_4 saturado proporciona un camino de baja resistencia desde V_{CC} hasta la salida; por tanto, un nivel BAJO en la entrada da lugar a un nivel ALTO en la salida. También, en este mismo instante, el emisor de Q_2 está a potencial de tierra, manteniendo bloqueado a Q_3.

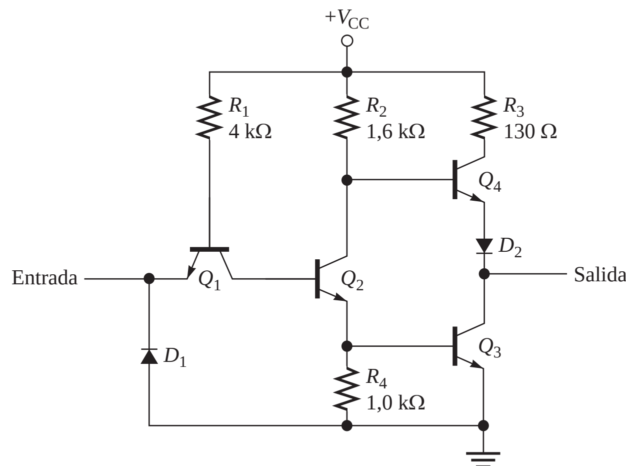


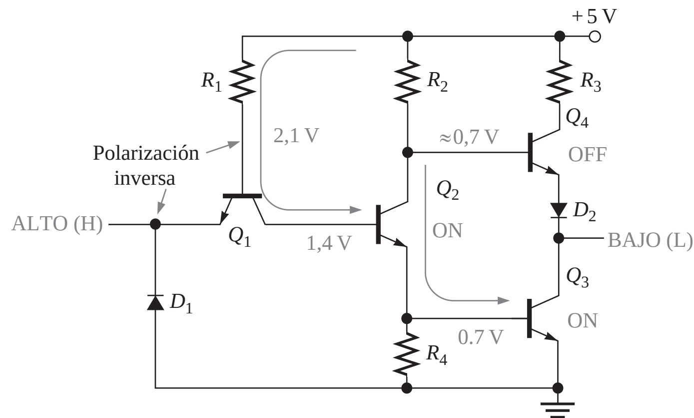
FIGURA 14.27 Circuito inversor TTL estándar.

El diodo D_1 del circuito TTL evita que los posibles picos negativos de tensión en la entrada pueden dañar a Q_1. El diodo D_2 asegura que Q_4 quede bloqueado cuando Q_2 conduce (entrada a nivel ALTO). En estas condiciones, la tensión de colector de Q_2 es igual a la tensión base-emisor V_{BE} de Q_3, más la tensión colector-emisor V_{CE} de Q_2. El diodo D_2 proporciona una caída adicional equivalente a V_{BE}, en serie con la unión base-emisor de Q_4, para asegurar su bloqueo cuando Q_2 conduce.

El funcionamiento del inversor TTL para los dos estados de entrada se ilustra en la Figura 14.28. En el circuito de la parte (a), la base de Q_1 está a 2,1 V respecto de masa, por lo que Q_2 y Q_3 se saturan. En el circuito de la parte (b), la base de Q_1 está aproximadamente a 0,7 V con respecto a tierra, lo que no es suficiente para poder saturar a Q_2 y Q_3.

Puerta NAND TTL

En la Figura 14.29 se muestra una puerta NAND TTL estándar de 2 entradas. Básicamente, es igual al circuito inversor, excepto en que Q_1 tiene un emisor de entrada adicional. En la tecnología TTL se utilizan transis-



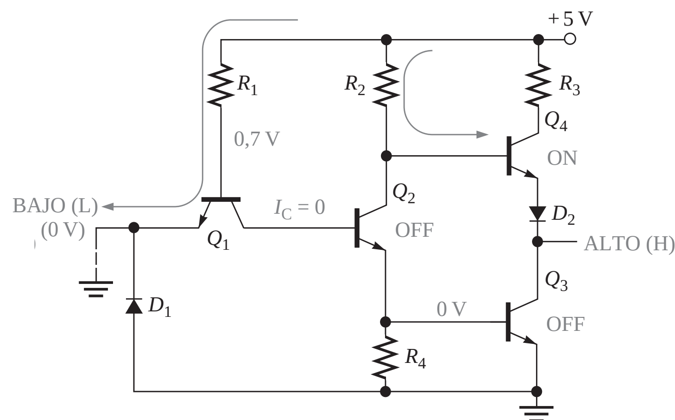


FIGURA 14.28 Funcionamiento de un inversor TTL.

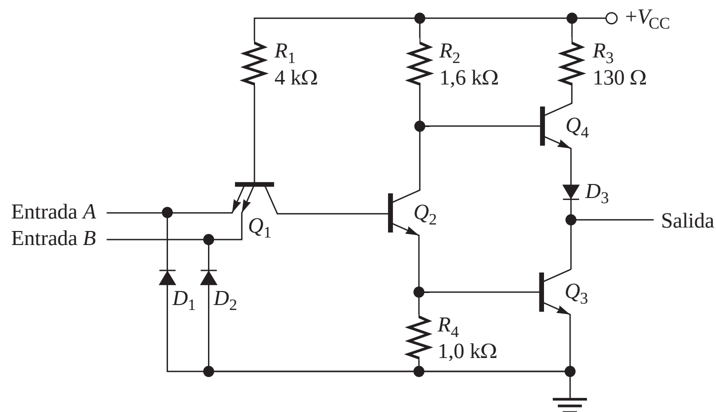


FIGURA 14.29 Circuito de una puerta NAND TTL.

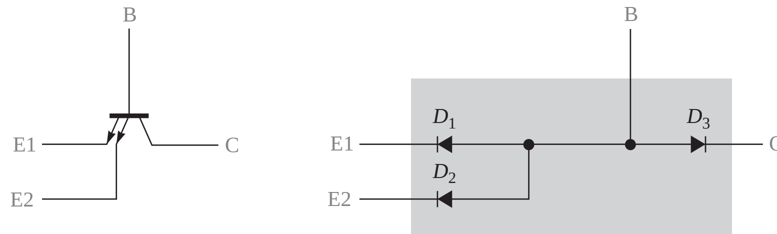


FIGURA 14.30 Equivalente con diodos de un transistor de múltiples emisores TTL.

tores con múltiples emisores para los dispositivos de entrada. Estos transistores de múltiples emisores pueden compararse con la disposición de diodos mostrada en la Figura 14.30.

Quizás sea más fácil entender el funcionamiento de este circuito reemplazando el transistor Q_1 de la Figura 14.29 por la combinación de diodos de la Figura 14.30. Un nivel BAJO en la entrada A o en la entrada B polariza en directa al respectivo diodo y en inversa a D_3 (unión base-colector de Q_1). Esto hace que Q_2 no conduzca, y da lugar a un nivel ALTO de salida, de la misma forma que se ha descrito para el inversor TTL. Por supuesto, un nivel BAJO en ambas entradas producirá este mismo resultado.

Un nivel ALTO en ambas entradas polariza en inversa a ambos diodos y en directa al diodo D_3 (unión base-colector de Q_1). Esto hace que Q_2 entre en conducción y da lugar a un nivel BAJO de salida, del mismo modo que se ha descrito para el inversor TTL. Debería reconocer este modo de operación como el de la función NAND: la salida es un nivel BAJO si y sólo si todas las entradas están a nivel ALTO.

Puertas en colector abierto

Todas las puertas TTL descritas en las secciones anteriores tienen el circuito de salida *totem-pole*. Los circuitos integrados TTL disponen de otro de tipo de salida, que es la salida en ***colector abierto.*** Este tipo de salida es comparable a la salida en drenador abierto de los CMOS. En la Figura 14.31(a) se presenta un inversor TTL estándar con salida en colector abierto. Otros tipos de puertas también están disponibles con este tipo de salidas.

Observe que la salida es el colector del transistor Q_3, sin nada conectado; de ahí el nombre de *colector abierto*. Para obtener los niveles lógicos ALTO y BAJO a la salida del circuito se conecta una resistencia de *pull-up* a V_{CC} desde el colector de Q_3, como muestra la Figura 14.31(b). Cuando Q_3 no conduce, la salida se lleva a V_{CC} a través de la resistencia externa. Cuando Q_3 se satura, la salida se lleva a un potencial próximo a tierra a través del transistor saturado.

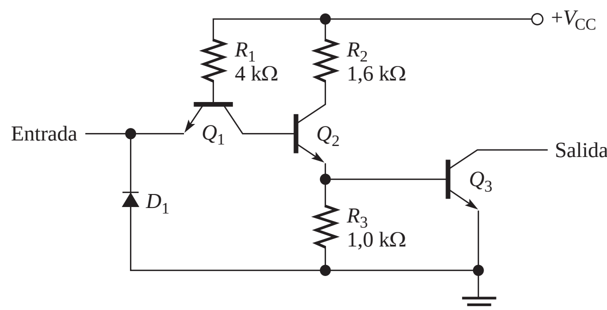
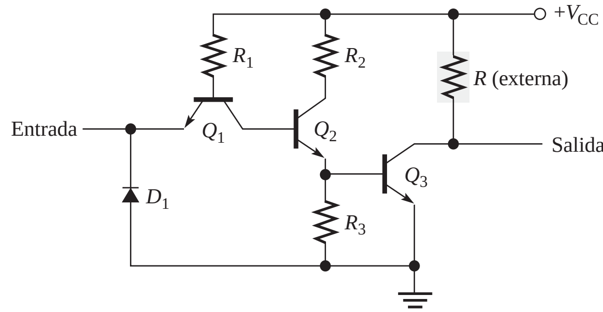


FIGURA 14.31 Inversor TTL estándar con salida en colector abierto. (*Continúa*)



(b) Con resistencia de *pull-up* externa

FIGURA 14.31 (*Continuación*)

El símbolo estándar ANSI/IEEE que designa la salida en colector abierto para un inversor se muestra en la Figura 14.32, y es el mismo que para una salida en drenador abierto.

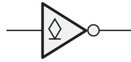


FIGURA 14.32 Símbolo de colector abierto en un inversor.

Puertas TTL con salida triestado

La Figura 14.33 muestra el circuito básico de un inversor triestado TTL. Cuando la entrada de habilitación está a nivel BAJO, Q_2 no conduce y el circuito de salida funciona en la configuración *totem-pole* normal, en la que el estado de salida depende del estado de entrada. Cuando la entrada de habilitación está a nivel ALTO, Q_2 conduce. Debido a ello, en el segundo emisor de Q_1 se produce un nivel BAJO, haciendo que Q_3 y Q_5 se bloqueen y el diodo D_1 se polarice en directa, lo que hace que Q_4 se bloquee también. Cuando ambos transistores *totem-pole* se bloquean, actúan como un circuito abierto y la salida está por completo desconectada de la circuitería interna, como muestra la Figura 14.34.

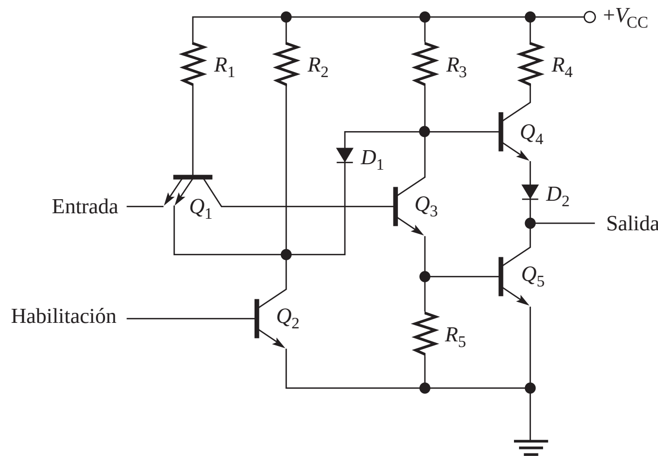


FIGURA 14.33 Circuito inversor de tres estados básico.

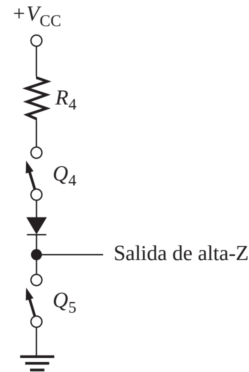


FIGURA 14.34 Circuito equivalente de la salida triestado en el estado de alta-Z.

TTL Schottky

Anteriormente se ha estudiado el circuito básico de la puerta NAND TTL. Es un sumidero de corriente que absorbe corriente de la carga cuando el estado de salida es un nivel BAJO, y suministra una corriente despreciable a la carga cuando el estado de salida es ALTO. La mayor parte de la lógica TTL utilizada actualmente es de tipo TTL Schottky, que proporciona un muy rápido tiempo de conmutación mediante la incorporación de diodos *Schottky*, que evitan que los transistores entren en saturación, disminuyendo el tiempo que tarda el transistor en entrar y salir de conducción. La Figura 14.35 muestra un circuito de puerta Schottky. Observe los símbolos de los transistores Schottky y de los diodos Schottky. Los dispositivos Schottky se designan mediante mediante una *S* en su identificador de dispositivo, como por ejemplo 74S00. Otros tipos de TTL Schottky son la lógica TTL Schottky de bajo consumo, que se designa mediante las letras LS; la lógica Schottky avanzada, que se designa mediante AS; Schottky de bajo consumo avanzada, que se especifica mediante ALS; y la serie rápida, que se designa mediante la letra F.

REVISIÓN DE LA SECCIÓN 14.3

1. Verdadero o falso: un BJT *npn* se satura cuando la base es más negativa que el emisor.
2. En términos de conmutación, ¿qué representan los estados de saturación y bloqueo de un BJT?
3. ¿Cuáles son los dos principales tipos de salida en los circuitos TTL?
4. Explicar en qué difiere la lógica de tres estados de la lógica normal de dos estados.

14.4 CONSIDERACIONES PRÁCTICAS SOBRE EL USO DE TTL

Aunque CMOS es la tecnología de circuitos integrados predominante en la industria y en las aplicaciones comerciales, la tecnología TTL todavía se emplea. En las aplicaciones para la enseñanza, normalmente se prefiere la tecnología TTL porque no tiene las restricciones de manipulación que tiene la tecnología CMOS, debidas a las descargas electrostáticas. Por esto, vamos a examinar varias consideraciones prácticas para el uso y aplicación de los circuitos TTL, utilizando la lógica TTL estándar como ilustración.

Al finalizar esta sección, el lector deberá ser capaz de:

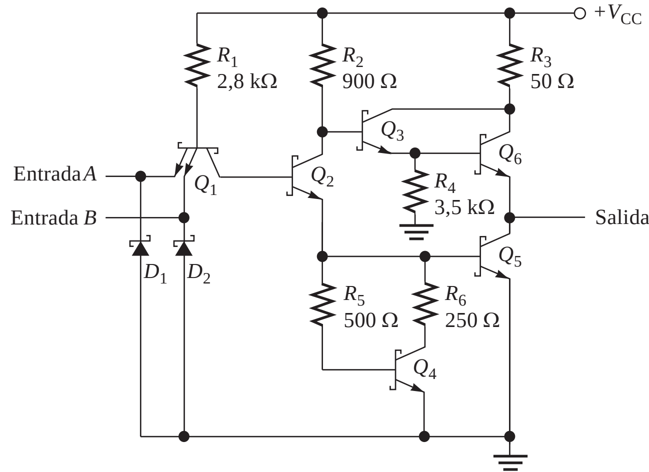


FIGURA 14.35 Puerta NAND TTL Schottky.

■ Describir los conceptos de fuente y sumidero de corriente. ■ Utilizar un circuito en colector abierto para la operación AND-cableada. ■ Describir los efectos de conectar dos o más salidas *totem-pole.* ■ Utilizar las puertas en colector abierto para excitar diodos LED y lámparas. ■ Explicar qué se hace con las entradas TTL no utilizadas.

Fuente y sumidero de corriente

En la Sección 14.1 se han introducido los conceptos de fuente y sumidero de corriente. Ahora que ya estamos familiarizados con las configuraciones de los circuitos con salida *totem-pole* utilizados en la lógica TTL, vamos a ver más en detalle la operación como fuente y sumidero de corriente.

La Figura 14.36 muestra un inversor TTL estándar, con una salida *totem-pole* conectada a la entrada de otro inversor TTL. Cuando la puerta excitadora tiene un estado de salida ALTO, actúa como fuente de corriente para la carga, como muestra la Figura 14.36(a). La entrada de la puerta de carga es como un diodo polarizado en inversa, por lo que la carga prácticamente no requiere corriente. Realmente, puesto que es una entrada no ideal, habrá un máximo de 40 μA de corriente desde la salida *totem-po*le del excitador a la entrada de la puerta de carga.

Cuando la puerta excitadora está en el estado de salida BAJO, el excitador actúa como sumidero de corriente de la carga, como muestra la Figura 14.36(b). Esta corriente es de 1,6 mA como máximo para los dispositivos TTL estándar y en las **hojas de características** se indica con un valor negativo, puesto que *sale* de la entrada.

EJEMPLO 14.3

Cuando una puerta NAND TTL estándar excita cinco entradas TTL, ¿cuánta corriente entrega (fuente) la salida del excitador y cuánta absorbe (sumidero)? (Consultar la Figura 14.36.)

Solución La corriente total de fuente es (para el estado de salida ALTO):

$$I_{\text{IH(máx)}} = 40\,\mu\text{A por entrada}$$

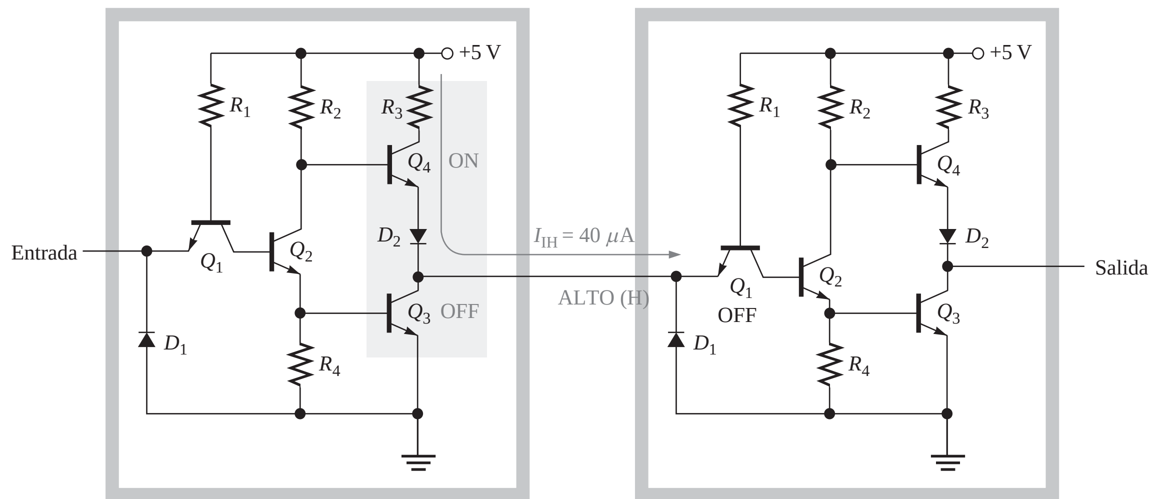
$$I_{\text{T(fuente)}} = (5\text{ entradas})(40\,\mu\text{A por entrada}) = 5(40\,\mu\text{A}) = \mathbf{200\,\mu A}$$

La corriente total absorbida (en el estado de salida BAJO) es:

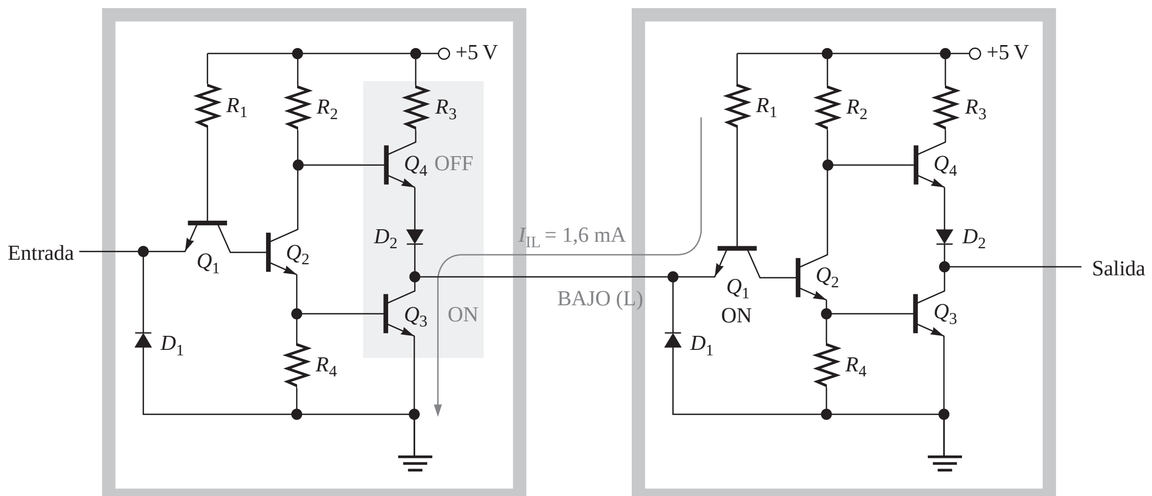
$$I_{\text{IH(máx)}} = -1{,}6\,\text{mA por entrada}$$

$$I_{\text{T(sumidero)}} = (5\ \text{entradas})(-1{,}6\,\text{mA}/\text{entrada}) = 5(-1{,}6\,\text{mA}) = \mathbf{-8{,}0\,mA}$$

Problema relacionado Repetir los cálculos para una puerta NAND TTL LS. Utilizar una hoja de características.



(a) Fuente de corriente (el valor de I_{IH} es máximo)



(b) Sumidero de corriente (el valor de I_{IL} es máximo)

FIGURA 14.36 Operación de un circuito TTL como fuente y sumidero de corriente.

EJEMPLO 14.4

Utilizando hojas de características, determinar el *fan-out* de la puerta NAND 7400.

Solución De acuerdo con la hoja de características, los parámetros de corriente son:

$$I_{IH(máx)} = 40\,\mu A \qquad I_{OH(máx)} = -400\,\mu A$$
$$I_{IL(máx)} = -1{,}6\,mA \qquad I_{OL(máx)} = 16\,mA$$

El *fan-o*ut para el estado de salida ALTO se obtiene del siguiente modo: la corriente $I_{OH(máx)}$ es la corriente máxima que la puerta puede entregar a la carga. Cada entrada de carga requiere una $I_{IH(máx)}$ de 40 μA. El *fan-out* del estado ALTO es:

$$\left|\frac{I_{OH(máx)}}{I_{IH(máx)}}\right| = \frac{400\,\mu A}{40\,\mu A} = \mathbf{10}$$

Para el estado de salida a nivel BAJO, el *fan-out* se calcula como sigue: $I_{OL(máx)}$ es la máxima corriente que la puerta puede absorber. Cada entrada de carga requiere una $I_{IL(máx)}$ de −1,6 mA. El *fan-out* del estado BAJO es:

$$\left|\frac{I_{OL(máx)}}{I_{IL(máx)}}\right| = \frac{16\,mA}{1{,}6\,mA} = \mathbf{10}$$

En este caso, el *fan-out* del estado ALTO y del estado BAJO son iguales.

Problema relacionado Determinar el *fan-out* de una puerta NAND 74LS00.

Utilización de puertas en colector abierto para la operación AND-cableada

Las salidas de las puertas en colector abierto se pueden conectar juntas para formar lo que se denomina una configuración *AND-cableada.* La Figura 14.37 ilustra cómo se conectan cuatro inversores para generar una puerta negativa-AND de 4 entradas. En todos los circuitos de AND-cableada, se requiere una resistencia de *pull-up* externa, R_p.

Cuando una (o más) de las entradas de los inversores están a nivel alto, la salida X pasa a nivel BAJO, puesto que un transistor de salida conduce y actúa como un interruptor cerrado conectado a tierra, como ilustra la Figura 14.38(a). En este caso, sólo la entrada de un inversor está a nivel ALTO, pero es suficiente para forzar la salida al nivel BAJO por medio de la saturación del transistor Q_1, tal y como se indica.

Para que la salida X sea un nivel ALTO, *todas* las entradas de los inversores deben estar a nivel BAJO, de modo que los transistores de salida en colector abierto estén bloqueados, como se indica en la Figura 14.38(b). Cuando se produce esta condición, se fuerza a la salida X al nivel ALTO por medio de la resistencia de *pull-up*. Luego la salida X es un nivel ALTO sólo cuando *todas* las entradas están a nivel BAJO. Por tanto, se trata de una función negativa-AND, como expresa la siguiente ecuación:

$$X = \overline{A}\overline{B}\overline{C}\overline{D}$$

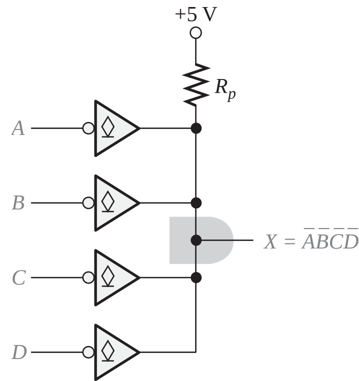


FIGURA 14.37 Configuración AND-cableada con cuatro inversores.

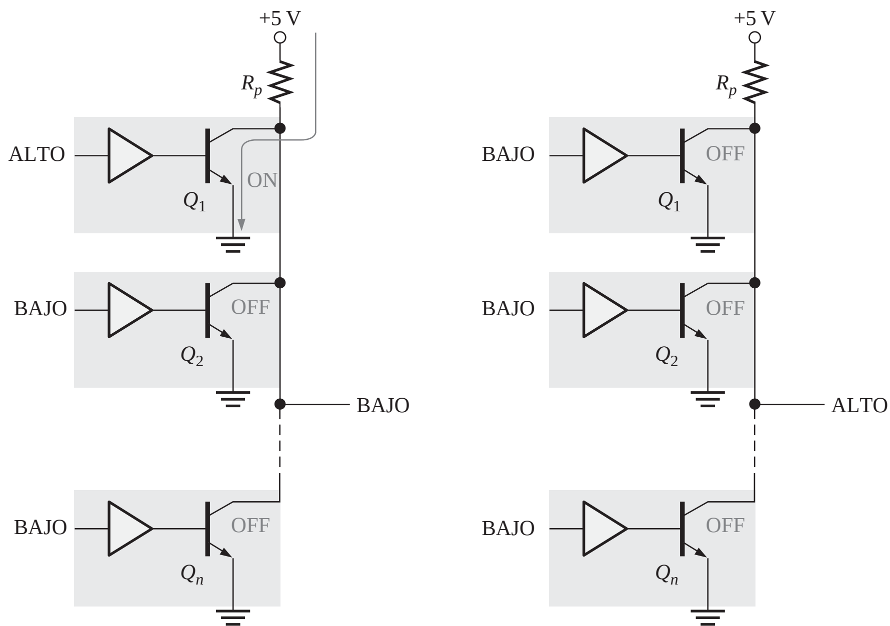


FIGURA 14.38 Operación negativa-AND con inversores con salida en colector abierto cableada.

EJEMPLO 14.5

Escribir la expresión de salida de la configuración AND-cableada con puertas AND en colector abierto de la Figura 14.39.

Solución La expresión de salida es:

$$X = ABCDEFGH$$

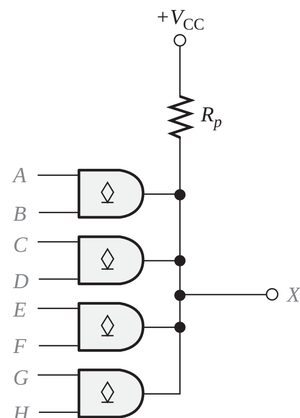


FIGURA 14.39

La conexión AND-cableada de las cuatro puertas AND de 2 entradas da lugar a una puerta AND de 8 entradas.

Problema relacionado Determinar la expresión de salida si, en la Figura 14.39, se usan puertas NAND.

EJEMPLO 14.6

Tres puertas AND en colector abierto se conectan en una configuración AND-cableada como la mostrada en la Figura 14.40. Suponer que el circuito AND-cableada se excita mediante cuatro entradas TTL estándar (−1,6 mA cada una).

(a) Escribir la expresión lógica de *X*.
(b) Determinar el valor mínimo de R_p, si $I_{OL(máx)}$ para cada puerta es 30 mA y $V_{OL(máx)}$ es 0,4 V.

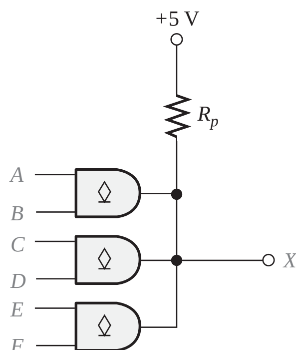


FIGURA 14.40

Solución

(a) $X = ABCDEF$
(b) 4(1,6 mA) = 6,4 mA

$$I_{R_p} = I_{\text{OL(máx)}} - 6{,}4\,\text{mA} = 30\ \text{mA} - 6{,}4\,\text{mA} = 23{,}6\,\text{mA}$$

$$R_p = \frac{V_{\text{CC}} - V_{\text{OL(máx)}}}{I_{R_p}} = \frac{5\,\text{V} - 0{,}4\ \text{V}}{23{,}6\,\text{mA}} = \mathbf{195\,\Omega}$$

Problema relacionado Diseñar el circuito AND-cableada para obtener una función AND de 10 entradas utilizando circuitos 74LS09, cuádruple puerta AND de 2 entradas.

Conexión de las salidas *totem-pole*

Las salidas *totem-pole* no se pueden conectar juntas, porque dicha conexión produce una corriente excesiva y daría lugar a daños en los dispositivos. Por ejemplo, en la Figura 14.41, cuando los transistores Q_1 del dispositivo A y Q_2 del dispositivo B conducen, la salida del dispositivo queda cortocircuitada a tierra a través del transistor Q_2 del dispositivo B.

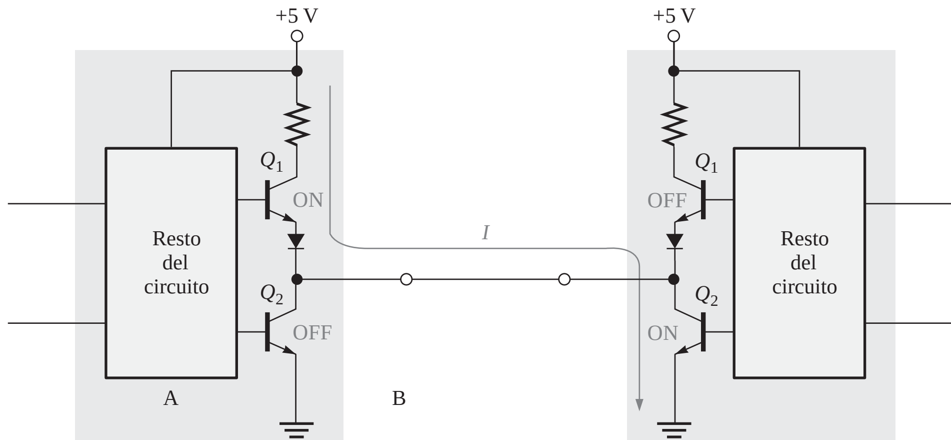


FIGURA 14.41 Salidas *totem-pole* conectadas juntas. Esta conexión puede generar una corriente excesiva a través del transistor Q_1 del dispositivo A y del transistor Q_2 del dispositivo B, y no debería emplearse nunca.

Buffer/excitador en colector abierto

Un circuito TTL con una salida *totem-pole* tiene limitada la cantidad de corriente que puede absorber en el estado BAJO ($I_{\text{OL(máx)}}$), a 16 mA para la lógica TTL estándar y a 8 mA para la lógica TTL LS. En muchas aplicaciones especiales, una puerta tiene que excitar a dispositivos externos tales como diodos LED, lámparas o relés, que pueden requerir más corriente que la que pueden proporcionar estos dispositivos.

Generalmente, para excitar diodos LED, lámparas o relés, se utilizan circuitos con salidas en colector abierto, debido a su mayor capacidad de manejo de corriente y tensión. Sin embargo, las salidas *totem-p*ole pueden utilizarse mientras que la corriente de salida requerida por el dispositivo externo no exceda el valor que el excitador TTL pueda absorber.

En una puerta TTL en colector abierto, el colector del transistor de salida se conecta al **LED** o a la lámpara incandescente, como muestra la Figura 14.42. En la parte (a), se utiliza una resistencia de limitación para mantener la corriente por debajo de la corriente máxima del LED. Cuando la salida de la puerta es un nivel BAJO, el transistor de salida opera como sumidero de corriente, y el LED se enciende. El LED se apaga cuan-

do el transistor de salida no conduce y la salida está a nivel alto. Una puerta *buffer* en colector abierto típica puede absorber hasta 40 mA. En la parte (b) de la figura, la lámpara no requiere una resistencia de limitación, ya que el filamento es resistivo. Típicamente, en colector abierto pueden utilizarse hasta +30 V, dependiendo de la familia lógica en particular.

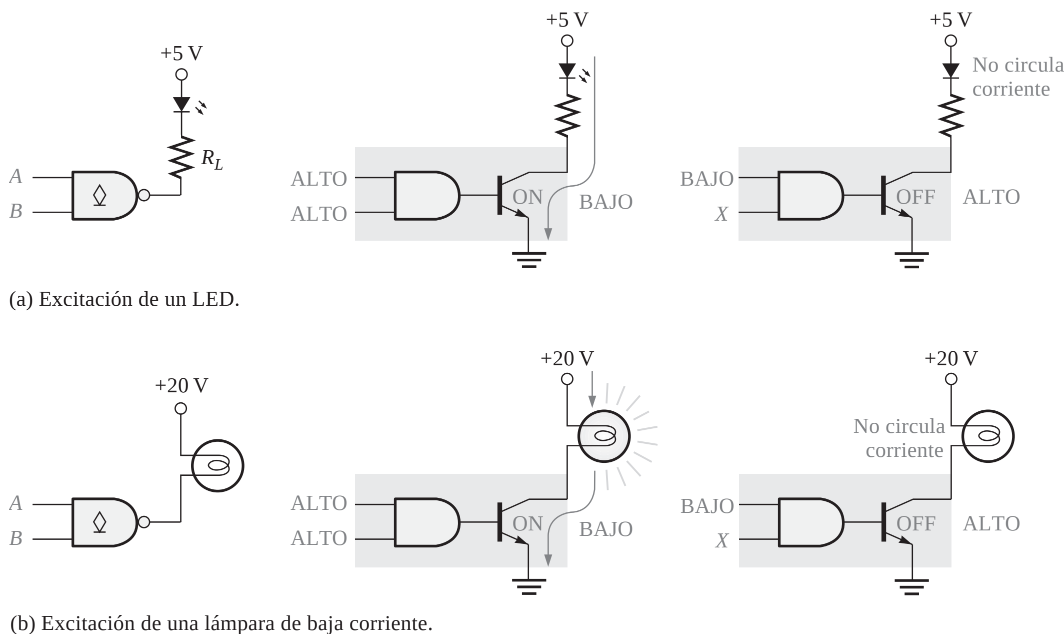


FIGURA 14.42 Aplicaciones de los excitadores en colector abierto.

EJEMPLO 14.7

Determinar la resistencia de limitación R_L en el circuito en colector abierto de la Figura 14.43, si la corriente del LED debe ser 20 mA. Suponer una caída de tensión de 1,5 V en el LED cuando está polarizado en directa y una tensión de salida para el estado BAJO de 0,1 V en la salida de la puerta.

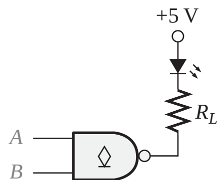


FIGURA 14.43

Solución

$$V_{R_L} = 5\,\text{V} - 1{,}5\,\text{V} - 0{,}1\ \text{V} = 3{,}4\ \text{V}$$

$$R_L = \frac{V_{R_L}}{I} = \frac{3{,}4\,\text{V}}{20\ \text{mA}} = \mathbf{170\,\Omega}$$

Problema relacionado Determinar el valor de la resistencia de limitación R_L, si el LED requiere 35 mA.

Entradas TTL no utilizadas

Una entrada desconectada de una puerta TTL actúa como un nivel lógico ALTO, ya que una entrada en abierto da lugar a una unión de emisor polarizada en inversa en el transistor de entrada, al igual que un nivel ALTO. En la Figura 14.44 se ilustra este efecto. Sin embargo, debido a la sensibilidad al ruido, es mejor no dejar las entradas no utilizadas desconectadas (en abierto). Existen varias alternativas para tratar este tipo de entradas no utilizadas.

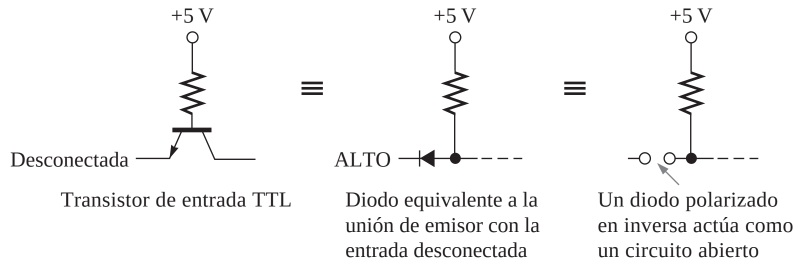
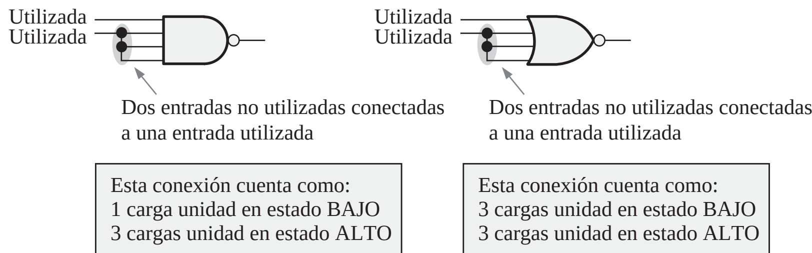


FIGURA 14.44 Comparación de una entrada TTL en abierto y una entrada a nivel ALTO.

Entradas unidas. El método más común de manipular las entradas no utilizadas de una puerta es conectarlas a una entrada que sí se use de la misma puerta. Para las puertas NAND y AND, todas las entradas unidas cuentan como una unidad de carga para el estado BAJO; pero en las puertas OR y NOR, cada entrada conectada a otra entrada cuenta como una carga adicional en el estado BAJO. En el estado ALTO, cada entrada unida cuenta como una unidad de carga adicional para todos los tipos de puertas TTL. En la Figura 14.45(a) se presentan dos ejemplos de conexión de dos entradas no utilizadas a una entrada que sí se utiliza.



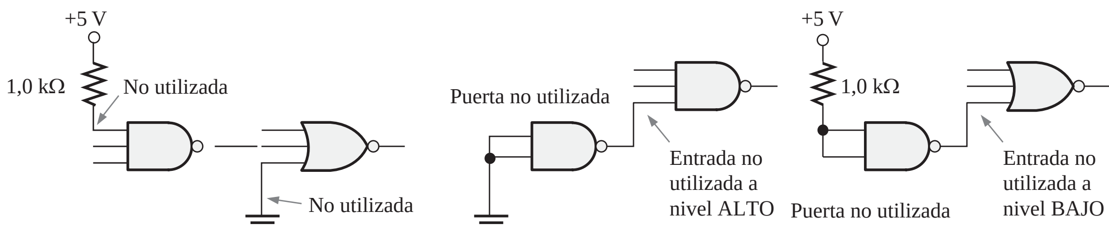


FIGURA 14.45 Métodos para tratar las entradas TTL no utilizadas.

Las puertas AND y NAND representan una única unidad de carga, independientemente de cuántas entradas se conecten juntas, mientras que para las puertas OR y NOR, cada entrada unida representa una unidad de carga. Esto se debe a que la puerta NAND utiliza un transistor de entrada de múltiples emisores, por lo que, independientemente de cuántas entradas estén a nivel BAJO, la corriente total para el estado BAJO está limitada a un valor fijo. La puerta NOR utiliza un transistor distinto para cada entrada, luego para el estado BAJO, la corriente total es la suma de las corrientes de todas las entradas unidas.

Entradas conectadas a V_{CC} o a tierra. Las entradas no utilizadas de las puertas AND y NAND se pueden conectar a V_{CC} a través de una resistencia de 1,0 kΩ. Esta conexión fuerza a las entradas no utilizadas a un nivel ALTO. Las entradas no utilizadas de las puertas OR y NOR se conectan a tierra. Estos métodos se ilustran en la Figura 14.45(b).

Entradas conectadas a una salida no utilizada. Un tercer método de terminación de las entradas no utilizadas puede ser adecuado en ciertos casos, cuando se dispone de una puerta o un inversor no utilizado. La salida de la puerta no utilizada debe estar a un nivel ALTO constante para las entradas AND y NAND no utilizadas, y a un nivel BAJO constante para las entradas OR y NOR, como muestra la Figura 14.45(c).

REVISIÓN DE LA SECCIÓN 14.4

1. ¿En qué estado de salida absorbe corriente de la carga un circuito TTL?
2. ¿Por qué un circuito TTL entrega menos corriente a una carga TTL que la que absorbe?
3. ¿Por qué los circuitos TTL con salidas *totem-pole* no se pueden conectar entre sí?
4. ¿Qué tipo de circuitos TTL deben utilizarse para obtener una configuración AND-cableada?
5. ¿Qué tipo de circuitos TTL debería utilizar para excitar una lámpara?
6. Una entrada TTL desconectada actúa como un nivel BAJO (verdadero o falso).

14.5 COMPARACIÓN DE LAS PRESTACIONES CMOS Y TTL

En esta sección, se van a comparar las principales prestaciones y características de funcionamiento de ciertas familias CMOS con las de las principales familias TTL y con BiCMOS.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Comparar los dispositivos TTL (bipolar), BiMOS y CMOS en términos de retardo de propagación, frecuencia de reloj máxima, disipación de potencia y capacidad de excitación.

En el pasado, las características superiores de los dispositivos TTL (bipolar) frente a los CMOS eran su relativamente alta velocidad y la capacidad de corriente de salida. Ahora, estas ventajas de la tecnología TTL han disminuido hasta el punto en que los dispositivos CMOS, a menudo, son iguales o superiores en muchas áreas y han llegado a ser la tecnología dominante de circuitos integrados, aunque la lógica TTL todavía está ampliamente disponible y en uso, como ya sabemos. Existe una familia de circuitos integrados, BiCMOS, que combina la lógica CMOS con la circuitería de salida TTL, en un esfuerzo por combinar las ventajas de ambas tecnologías.

La Tabla 14.1 proporciona una comparación de los parámetros de funcionamiento de varias familias lógicas de circuitos integrados.

REVISIÓN DE LA SECCIÓN 14.5

1. ¿Qué es un circuito BiCMOS?
2. En general, ¿cuál es la principal ventaja de la tecnología CMOS sobre la bipolar (TTL)?

	Bipolar (TTL)			BiCMOS	CMOS					
					5 V			3,3 V		
	F	LS	ALS	ABT	HC	AC	AHC	LV	LVC	ALVC
Velocidad Retardo de propagación de puerta, t_p (ns)	3,3	10	7	3,2	7	5	3,7	9	4,3	3
Frecuencia máxima de reloj (MHz)	145	33	45	150	50	160	170	90	100	150
Disipación de potencia/puerta Bipolar: 50% dc (mW)	6	2,2	1,4							
CMOS: reposo (μW)				17	2,75	0,55	2,75	1,6	0,8	0,8
Excitación de salida I_{OL} (mA)	20	8	8	64	4	24	8	12	24	24

TABLA 14.1 Comparación de los principales parámetros de funcionamiento de varias familias de circuitos integrados 74XX.

14.6 CIRCUITOS ECL

ECL (*Emitter-Coupled Logic*, lógica de emisor acoplado), al igual que la TTL, es una tecnológica bipolar. El circuito ECL típico está formado por un circuito de entrada amplificador diferencial, un circuito de polarización y salidas de tipo seguidor de emisor. ECL es mucho más rápida que TTL, ya que los transistores no funcionan en saturación y se emplea en aplicaciones de alta velocidad especializadas.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Explicar en qué difiere la tecnología ECL de TTL y CMOS. ■ Explicar las ventajas y desventajas de la lógica ECL.

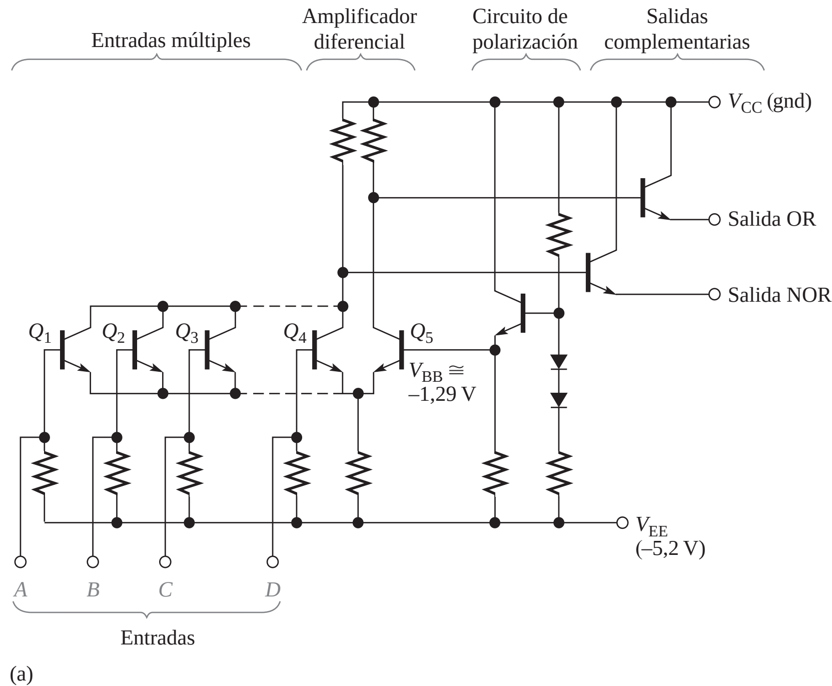
En la Figura 14.46(a) se presenta una puerta OR/NOR ***ECL***. Las salidas en seguidor de emisor proporcionan la función lógica OR y su complementaria NOR, como se indica en la Figura 14.46(b).

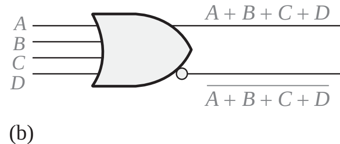
Debido a la baja impedancia de salida del seguidor de emisor y a la alta impedancia de entrada del amplificador diferencial, es posible trabajar con un alto *fan-out*. En este tipo de circuito, no es posible la saturación. Esta falta de saturación da lugar a un muy alto consumo de potencia y a recorridos de tensión limitados (menores de 1 V), pero permite la conmutación a alta frecuencia.

Normalmente, el pin V_{CC} se conecta a tierra y el pin V_{EE} se conecta a −5,2 V con respecto a la tensión de alimentación, para obtener un funcionamiento óptimo. Observe en la Figura 14.46(c) que la salida varía desde −1,75 V para el nivel BAJO a −0,9 V para el nivel ALTO, con respecto a masa. En la lógica positiva, un 1 es un nivel ALTO (menos negativo) y un 0 es un nivel BAJO (más negativo).

Margen de ruido

Como ya sabe, el margen de ruido de una puerta es la medida de su inmunidad a las fluctuaciones de tensión indeseadas (ruido). Típicamente, los circuitos ECL tienen márgenes de ruido comprendidos, aproximadamente, entre 0,2 V y 0,25 V. Estos márgenes son menores que en la lógica TTL, y hacen a la lógica ECL poco fiable en entornos de alto ruido.





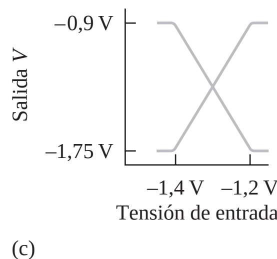


FIGURA 14.46 Circuito de una puerta OR/NOR ECL.

Comparación de ECL con TTL y CMOS

La Tabla 14.2 muestra una comparación de los parámetros clave de funcionamiento para las familias F y AHC, y ECL.

REVISIÓN DE LA SECCIÓN 14.6

1. ¿Cuál es la principal ventaja de ECL sobre TTL?
2. Citar dos desventajas de la lógica ECL respecto a la TTL.

	Bipolar (TTL) F	CMOS AHC	Bipolar (ECL)
Velocidad			
Retardo de propagación de puerta, t_p (ns)	3,3	3,7	0,22-1
Frecuencia máxima de reloj (MHz)	145	170	330-2800
Disipación de potencia/puerta			
Bipolar: 50% dc	8,9 mW		25 mW-73 mW
CMOS: reposo		2,5 μW	

TABLA 14.2 Comparación de los parámetros de funcionamiento de series ECL con las familias F y AHC.

14.7 PMOS, NMOS Y E^{2}CMOS

Los circuitos PMOS y NMOS se utilizan en muchas funciones LSI, tales como registros de desplazamiento, grandes memorias y microprocesadores. Su uso se debe al bajo consumo de potencia y la pequeña área de chip que requieren los transistores MOS. E^{2}CMOS se usa en dispositivos PLD reprogramables.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Describir una puerta PMOS básica. ■ Describir una puerta NMOS básica. ■ Describir una celda E^{2}CMOS básica.

PMOS

Una de las primeras tecnologías para producir circuitos **MOS** de alta densidad fue la **PMOS**. Utilizaba transistores MOS de canal-*p* en modo de acumulación para producir los elementos de puerta básicos. La Figura 14.47 muestra una puerta PMOS que genera la función NOR en lógica positiva.

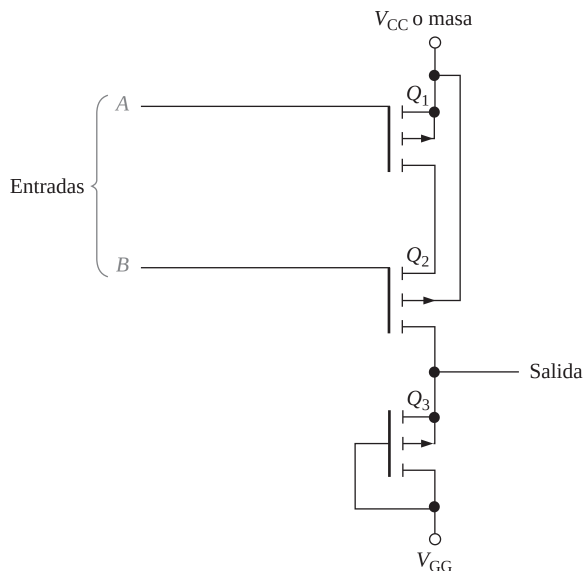


FIGURA 14.47 Puerta PMOS básica.

El funcionamiento de la puerta PMOS es el siguiente: la tensión de alimentación V_{GG} es una tensión negativa y V_{CC} es una tensión positiva o masa (0 V). El transistor Q_3 está polarizado permanentemente para crear una resistencia de drenador-fuente constante. Su único propósito es funcionar como resistencia limitadora de corriente. Si se aplica un nivel ALTO (V_{CC}) a la entrada A o B, entonces Q_1 o Q_2 no conducen, y la salida se fuerza a una tensión próxima a V_{GG}, lo que representa un nivel BAJO. Cuando se aplica una tensión a nivel BAJO (V_{GG}) a ambas entradas A y B, Q_1 y Q_2 entran en conducción. Esto hace que la salida pase a nivel ALTO (próximo a V_{CC}). Puesto que se produce un nivel BAJO de salida cuando cualquier entrada o ambas están a nivel ALTO, y un nivel ALTO de salida se produce sólo cuando todas las entradas están a nivel BAJO, tenemos una puerta NOR.

NMOS

Los dispositivos NMOS se han desarrollado a medida que la tecnología de procesamiento mejoraba. En los circuitos NMOS se utiliza el transistor MOS de canal-*n*, como muestra la Figura 14.48 para una puerta NAND y una puerta NOR.

En la Figura 14.48(a), Q_3 actúa como una resistencia para limitar la corriente. Cuando se aplica un nivel BAJO (V_{GG} o tierra) a una o ambas entradas, entonces al menos uno de los transistores (Q_1 o Q_2) no conduce, y la salida se fuerza a un nivel alto próximo a V_{CC}. Cuando se aplican niveles ALTOS (V_{CC}) a ambas entradas A y B, los transistores Q_1 y Q_2 conducen, y la salida se pone a nivel BAJO. Por supuesto, este funcionamiento identifica al circuito como una puerta NAND.

En la Figura 14.48(b), Q_3 actúa de nuevo como resistencia. Un nivel ALTO en cualquier entrada hace que Q_1 o Q_2 conduzcan, forzando la salida a nivel BAJO. Cuando ambas entradas están a nivel BAJO, ningún transistor conduce y se fuerza la salida al nivel ALTO.

E^2CMOS

La tecnología ***E^2CMOS*** (*Electrically Erasable CMOS*) está basada en una combinación de las tecnologías CMOS y NMOS y se utiliza en dispositivos programables como las PROM y dispositivos CPLD. Una celda E^2CMOS se construye a partir de un transistor MOS con una puerta flotante, que se carga o descarga externamente por medio de una pequeña corriente de programación. La Figura 14.49 presenta un esquema de este tipo de celda.

Cuando la puerta flotante se carga a un potencial positivo eliminando electrones, el transistor de detección se activa, almacenando un cero binario. Cuando la puerta flotante se carga a un potencial negativo incorporando electrones, el transistor de detección se desactiva, almacenando un 1 binario. La puerta de control regula el potencial de la puerta flotante. El transistor de paso aísla de la matriz al transistor de detección durante las operaciones de lectura y de escritura, que utilizan las líneas de palabra y de bit.

La celda se programa aplicando un impulso de programación a la puerta de control o a la línea de bit de una celda, después de seleccionarla por medio de una tensión en la línea de palabra. Durante el ciclo de programación, en primer lugar, se borra la celda aplicando una tensión a la puerta de control para hacer negativa a la puerta flotante. Esto hace que el transistor de detección se quede en el estado de *bloqueo*, almacenando un 1. Para almacenar un 0 en la celda, se aplica un impulso de escritura a su línea de bit. Esto hará que la puerta flotante se cargue hasta un punto en el que el transistor de detección se activa, almacenando un 0. El bit almacenado en la celda se lee detectando la presencia o ausencia de una pequeña corriente de celda en la línea de bit. Cuando se almacena un 1, no hay corriente de celda, porque el transistor de detección está desactivado. Cuando se almacena un 0, existe una pequeña corriente de celda debido a que el transistor de detección está activado. Una vez que se ha almacenado el bit en la celda, permanecerá indefinidamente, a menos que se borre o se escriba un nuevo dato en ella.

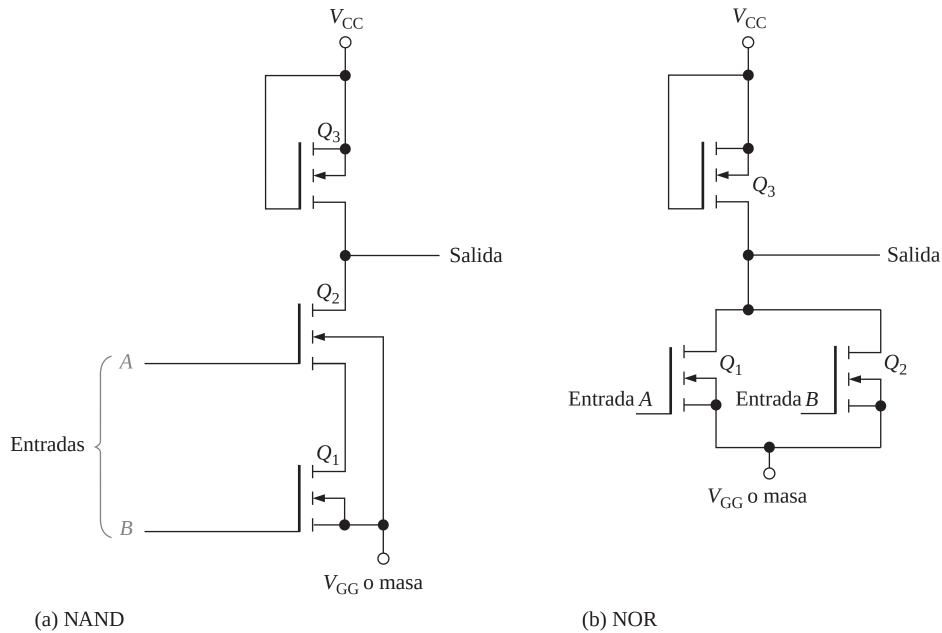


FIGURA 14.48 Dos puertas NMOS.

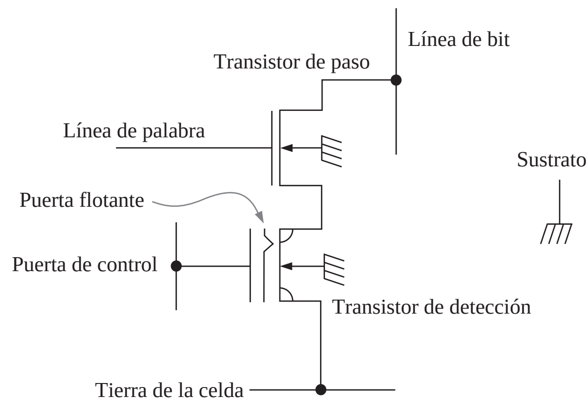


FIGURA 14.49 Celda E^2CMOS.

REVISIÓN DE LA SECCIÓN 14.7

1. ¿Cuál es la característica principal de la tecnología NMOS y PMOS en circuitos integrados?
2. ¿Cuál es el mecanismo para almacenar la carga en una celda E^2CMOS?

RESUMEN

■ *Fórmulas*

14.1 $V_{NH} = V_{OH(min)} - V_{IH(min)}$ Margen de ruido a nivel alto (H)

14.2 $V_{NL} = V_{IL(máx)} - V_{OL(máx)}$ Margen de ruido a nivel bajo (L)

14.3 $I_{CC} = \dfrac{I_{CCH} + I_{CCL}}{2}$ Corriente media continua de alimentación

14.4 $P_D = V_{CC} I_{CC}$ Disipación de potencia

- *Las salidas totem*-pole TTL no pueden conectarse juntas.
- Las salidas en colector abierto y en drenador abierto pueden conectarse en configuración AND-cableada.
- Los dispositivos CMOS ofrecen una disipación de potencia menor que cualquiera de la serie TTL.
- Un dispositivo TTL no es tan vulnerable a las descargas electrostáticas como un dispositivo CMOS.
- Debido a las descargas electrostáticas, los dispositivos CMOS deben manejarse con mucho cuidado.
- ECL es el tipo de circuito lógico más rápido.
- La tecnología E^2CMOS se utiliza en las PROM y otros dispositivos PLD.

PALABRAS CLAVE

Las palabras clave y otros términos que se han resaltado en negrita se encuentran en el glosario final del libro.

Carga unidad Una medida del *fan-out*. Una entrada de puerta representa una carga unidad para la salida de la puerta excitadora.

CMOS *Complementary Metal-Oxide Semiconductor*, metal-óxido semiconductor complementario. Un tipo de circuito de transistores que utiliza transistores MOSFET (*metal-oxide semiconductor field-effect*, transistor MOS de efecto de campo) de canal-*n* y canal-*p*.

Colector abierto Un tipo de salida en los circuitos TTL en el que el colector del transistor de salida se deja desconectado internamente y se encuentra disponible para conectarse a una carga externa que requiera una corriente o tensión relativamente altas.

Disipación de potencia El producto de la tensión continua de alimentación por la corriente continua de alimentación en un circuito electrónico.

ECL *Emitter-Coupled Logic*. Una clase de circuitos lógicos integrados que se implementan con transistores bipolares no saturados.

E^2CMOS *Electrically Erasable* CMOS, CMOS eléctricamente borrable. Tecnología de circuitos utilizada en dispositivos lógicos programables (PLD).

Fan-out El número de entradas de puerta equivalente de la misma serie de familias que puede excitar una puerta lógica.

Fuente de corriente Funcionamiento de un circuito cuando su salida entrega corriente a la carga.

Inmunidad al ruido Capacidad de un circuito para rechazar señales no deseadas.

Margen de ruido La diferencia entre la salida máxima a nivel BAJO de una puerta y el máximo nivel de entrada a nivel BAJO aceptable por una puerta equivalente; también, diferencia entre la salida mínima a nivel ALTO de una puerta y el mínimo nivel de entrada a nivel ALTO aceptable por una puerta equivalente. El margen de ruido algunas veces se expresa como un porcentaje de la tensión de alimentación continua.

Resistencia de *pull-u*p Resistencia que se utiliza para mantener un determinado punto de un circuito a nivel ALTO cuando se encuentra en estado inactivo.

Retardo de propagación Intervalo de tiempo que transcurre entre la transición de entrada y su correspondiente transición de salida en un circuito lógico.

Sumidero de corriente Funcionamiento de un circuito cuando su salida acepta corriente de la carga.

Totem pole Un tipo de salida de los circuitos TTL.

Triestado Un tipo de salida de los circuitos lógicos que exhibe tres estados: ALTO, BAJO y de alta impedancia (Alta-Z).

TTL *Transistor-Transistor Logic*, lógica transistor-transistor: un tipo de circuito integrado que utiliza transistores de unión bipolares.

AUTOTEST

Las respuestas se encuentran al final del capítulo.

1. Cuando la frecuencia de la señal de entrada a una puerta CMOS aumenta, la disipación media de potencia:
   (a) decrece **(b)** aumenta
   (c) no cambia **(d)** decrece exponencialmente
2. Los dispositivos CMOS son más fiables que los TTL en un entorno de alto ruido debido a su:
   (a) menor margen de ruido **(b)** capacitancia de entrada
   (c) mayor margen de ruido **(d)** menor disipación de potencia
3. Es necesario manejar correctamente un dispositivo CMOS por su:
   (a) frágil construcción
   (b) alta inmunidad al ruido
   (c) susceptibilidad a las descargas electrostáticas
   (d) baja disipación de potencia
4. ¿Cuál de los siguientes circuitos no es un circuito TTL?
   (a) 74F00 **(b)** 74AS00
   (c) 74HC00 **(d)** 74ALS00
5. Una entrada en abierto de una puerta NOR TTL:
   (a) actúa como un nivel BAJO
   (b) actúa como un nivel ALTO
   (c) debe conectarse a tierra
   (d) debería conectarse a V_{CC} a través de una resistencia
   (e) las respuestas (b) y (c) **(f)** las respuestas (a) y (c)
6. Una puerta TTL LS puede excitar un máximo de:
   (a) 20 unidades de carga
   (b) 10 unidades de carga
   (c) 40 unidades de carga
   (d) un número ilimitado de unidades de carga
7. Si dos entradas no utilizadas de una puerta TTL LS se conectan a una entrada que se está excitando con otra puerta TTL LS, el número total de unidades de cargas que esa puerta puede excitar es:
   (a) siete **(b)** ocho **(c)** diecisiete **(d)** ilimitado
8. La principal ventaja de ECL sobre TTL y CMOS es:
   (a) ECL es más barata
   (b) ECL consume menos potencia
   (c) ECL está disponible en una mayor variedad de tipos de circuitos
   (d) ECL es más rápida
9. ECL no se puede utilizar en:
   (a) entornos de alto ruido

(b) entornos húmedos

(c) aplicaciones de alta frecuencia

10. El mecanismo básico para almacenar un bit de datos en una celda E^2CMOS es:

(a) la puerta de control **(b)** el drenador flotante

(c) la puerta flotante **(d)** la corriente de celda

PROBLEMAS

Las respuestas a los problemas impares se encuentran al final del libro.

SECCIÓN 14.1 Parámetros y características de operación básicas

1. Una determinada puerta lógica tiene un $V_{OH(mín)}$ = 2,2 V, y excita a una puerta con un $V_{OH(mín)}$ = 2,5 V. ¿Son compatibles estas puertas para el funcionamiento en estado ALTO? ¿Por qué?

2. Una determinada puerta lógica tiene un $V_{OL(máx)}$ = 0,45 V y excita a una puerta con un $V_{IL(máx)}$ = 0,75 V ¿Son compatibles estas puertas para el funcionamiento en estado BAJO? ¿Por qué?

3. Una puerta TTL tiene los siguiente valores de nivel de tensión: $V_{IH(mín)}$ = 2,25 V, $V_{IL(máx)}$ = 0,65 V. Suponer que la está excitando una puerta con $V_{OH(mín)}$ = 2,4 V y $V_{OL(máx)}$ = 0,4 V, ¿cuáles son los márgenes de ruido para los niveles ALTO y BAJO?

4. ¿Cuál es la máxima amplitud de los picos de ruido que se puede tolerar en las entradas de la puerta del Problema 3, para el estado ALTO y el estado BAJO?

5. En la Tabla 14.3 se facilitan las especificaciones de tensión de tres tipos de puertas lógicas. Seleccionar la puerta que se debería utilizar en un entorno industrial de alto ruido.

	$V_{OH(mín)}$	$V_{OL(máx)}$	$V_{IH(mín)}$	$V_{IL(máx)}$
Puerta *A*	2,4 V	0,4 V	2 V	0,8 V
Puerta *B*	3,5 V	0,2 V	2,5 V	0,6 V
Puerta *C*	4,2 V	0,2 V	3,2 V	0,8 V

TABLA 14.3

6. Por una determinada puerta circula una corriente continua de alimentación procedente de una fuente de +5V y 2 mA en estado BAJO, y de 3,5 mA en estado ALTO. ¿Cuál es la disipación de potencia en el estado BAJO? ¿Cuál es la disipación de potencia en el estado ALTO? Suponiendo un ciclo de trabajo del 50%, ¿cuál es la disipación media de potencia?

7. Cada una de las puertas del circuito de la Figura 14.50 tiene un t_{PHL} y un t_{PLH} de 4 ns. Si se aplica a la entrada un impulso positivo como el indicado, ¿cuánto tiempo tardará en aparecer el impulso en la salida?

8. Para una determinada puerta, t_{PLH} = 3 ns y t_{PHL} = 2 ns. ¿Cuál es el retardo medio de propagación?

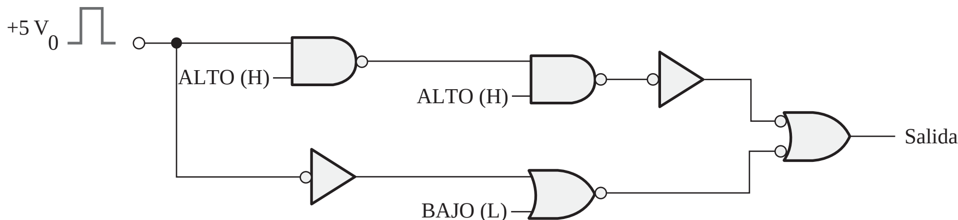


FIGURA 14.50

9. En la Tabla 14.4 se enumeran los parámetros de tres tipos de puertas. Basando su decisión en el producto velocidad-potencia, ¿cuál de ellas seleccionaría para obtener las máximas prestaciones?

	t_{PLH}	t_{PHL}	P_D
Puerta *A*	1 ns	1,2 ns	15 mW
Puerta *B*	5 ns	4 ns	8 mW
Puerta *C*	10 ns	10 ns	0,5 mW

TABLA 14.4

10. ¿Qué puerta de la Tabla 14.4 seleccionaría si deseara que la puerta funcionase a la frecuencia más alta posible?

11. Una puerta TTL estándar tiene un *fan-out* de 10. ¿Está sobrecargada alguna de las puertas de la Figura 14.51? Si la respuesta es afirmativa, ¿cuáles?

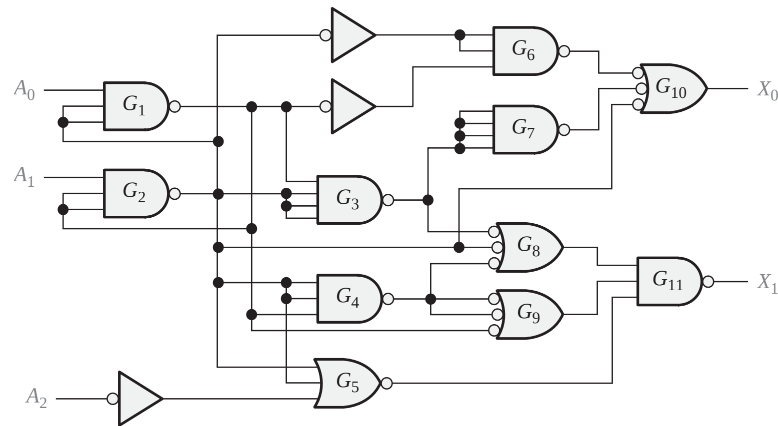


FIGURA 14.51

12. ¿Que red de puertas CMOS de la Figura 14.52 puede a trabajar a la frecuencia más alta?

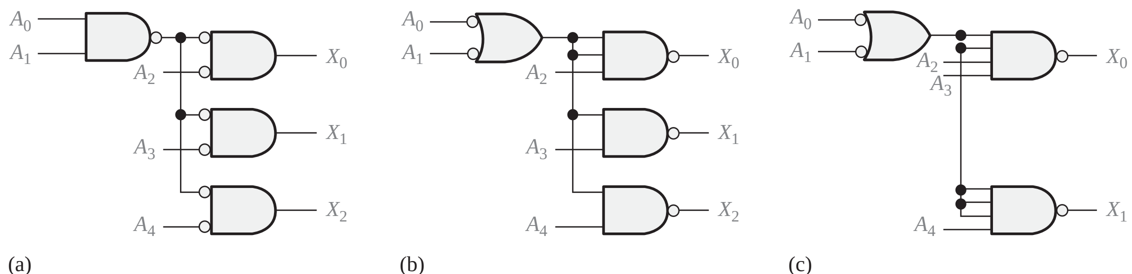


FIGURA 14.52

SECCIÓN 14.2 Circuitos CMOS

13. Determinar el estado (saturado o no, *on* u *off*) de cada MOSFET de la Figura 14.53.

14. La red de puertas CMOS de la Figura 14.54 está incompleta. Indicar los cambios que deberían hacerse.

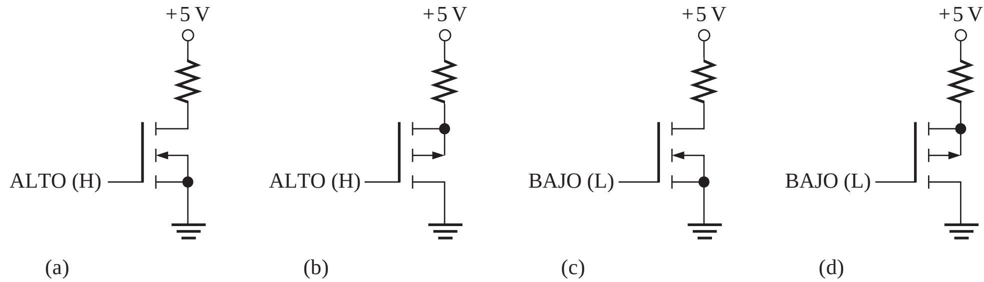


FIGURA 14.53

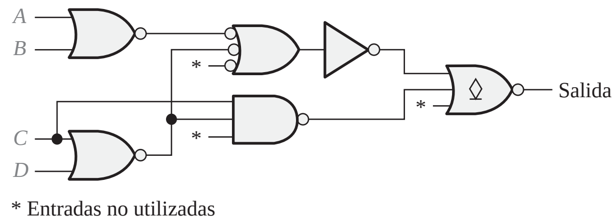


FIGURA 14.54

15. Diseñar un circuito, utilizando las apropiadas puertas lógicas y/o inversores CMOS, en el que las señales procedentes de cuatro diferentes fuentes puedan conectarse a una línea común en diferentes instantes, sin interferir entre ellas.

SECCIÓN 14.3 **Circuitos TTL**

16. Determinar cuáles BJT de la Figura 14.55 están saturados y cuáles no.

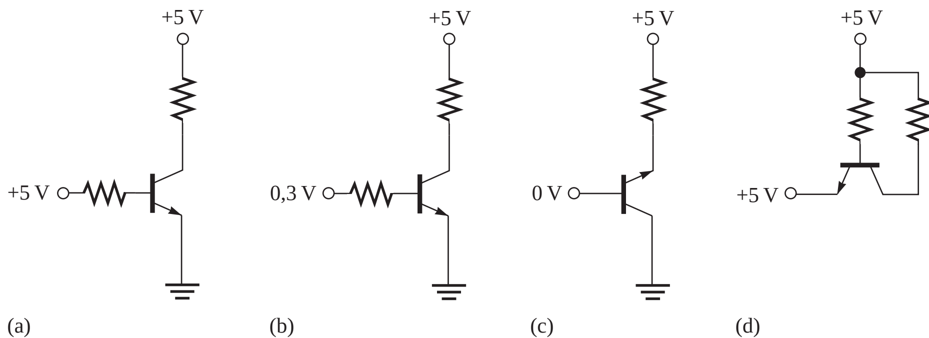


FIGURA 14.55

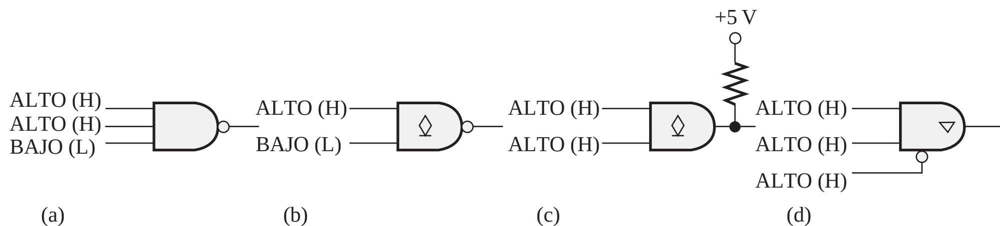


FIGURA 14.56

17. Determinar el estado de salida de cada una de las puertas TTL de la Figura 14.56.

18. La red de puertas TTL de la Figura 14.57 está incompleta. Indicar los cambios que habría que realizar.

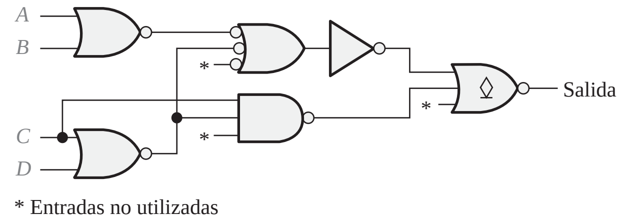


FIGURA 14.57

SECCIÓN 14.4 Consideraciones prácticas sobre el uso de TTL

19. Determinar el nivel de salida de cada puerta TTL de la Figura 14.58.

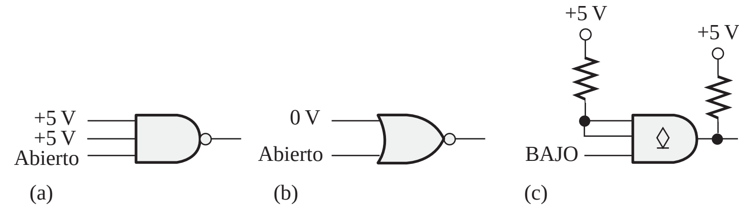


FIGURA 14.58

20. Para cada parte de la Figura 14.59, decir si la puerta excitadora actúa como fuente o sumidero de corriente. Especificar en cada caso la corriente máxima que sale o entra en la salida de la puerta o puertas excitadoras. Todas las puertas son TTL estándar.

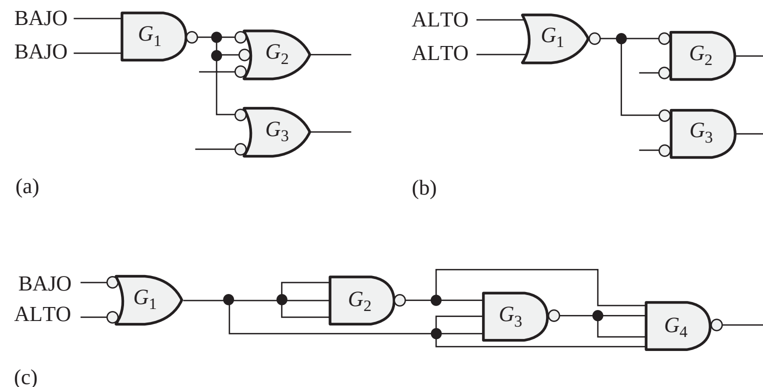


FIGURA 14.59

21. Utilizar inversores en colector abierto para implementar las siguientes expresiones lógicas:

(a) $X = \overline{A}\overline{B}\overline{C}$

(b) $X = A\overline{B}C\overline{D}$

(c) $X = ABC\overline{D}\overline{E}\overline{F}$

22. Escribir la expresión lógica para cada uno de los circuitos de la Figura 14.60.

23. Determinar el valor mínimo de la resistencia de *pull-up* de cada circuito de la Figura 14.60, si $I_{OL(máx)}$ = 40 mA y $V_{OL(máx)}$ = 0,25 V para cada puerta. Suponer que se excitan diez unidades de carga TTL estándar a partir de la salida *X* y que la tensión de alimentación es de 5 V.

24. Un cierto relé requiere 60 mA. Diseñar una forma de excitar al relé utilizando puertas NAND en colector abierto con una $I_{OL(máx)}$ = 40 mA.

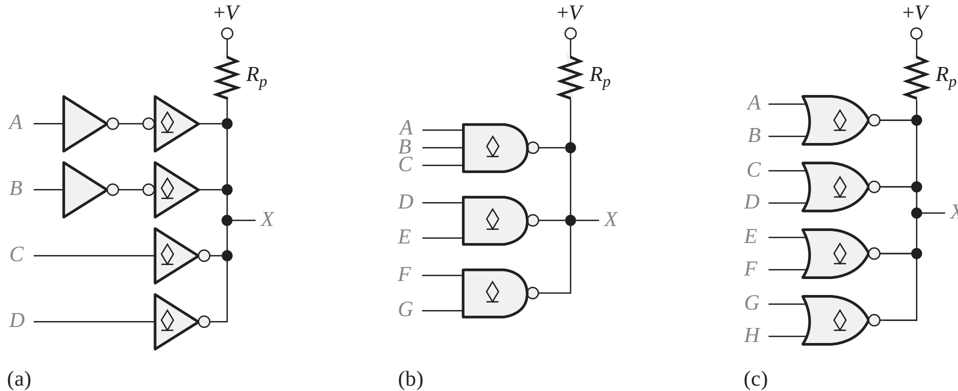


FIGURA 14.60

SECCIÓN 14.5 Comparación de las prestaciones CMOS y TTL

25. Seleccionar la familia de circuitos integrados con el mejor producto velocidad-potencia de la Tabla 14.1.

26. Determinar a partir de la Tabla 14.1 la familia lógica más apropiada para cada uno de los siguientes requisitos:

(a) retardo de propagación más corto

(b) velocidad de basculación de *flip-flop* más rápida

(c) disipación de potencia más baja

(d) mejor compromiso entre velocidad y potencia para una puerta lógica.

27. Determinar el retardo de propagación total desde cada entrada hasta cada salida para cada circuito de la Figura 14.61.

28. Uno de los *flip-flops* de la Figura 14.62 puede tener una salida errática, ¿cuál es y por qué?

SECCIÓN 14.6 Circuitos ECL

29. ¿Cuál es la diferencia básica entre los circuitos ECL y los circuitos TTL?

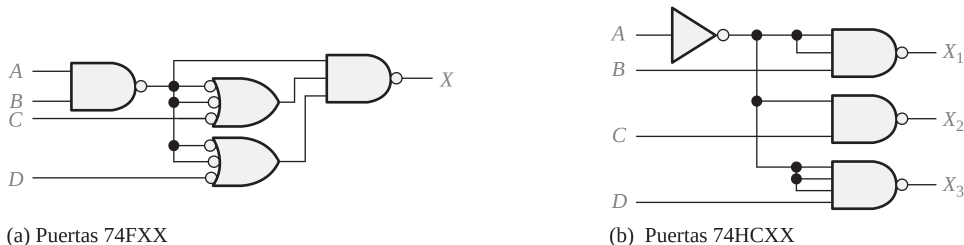
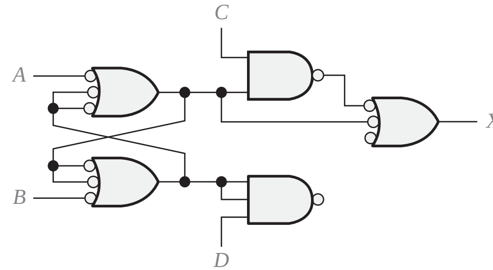


FIGURA 14.61 (*Continúa*)



(c) Puertas 74AHCXX

FIGURA 14.61 (*Continuación*)

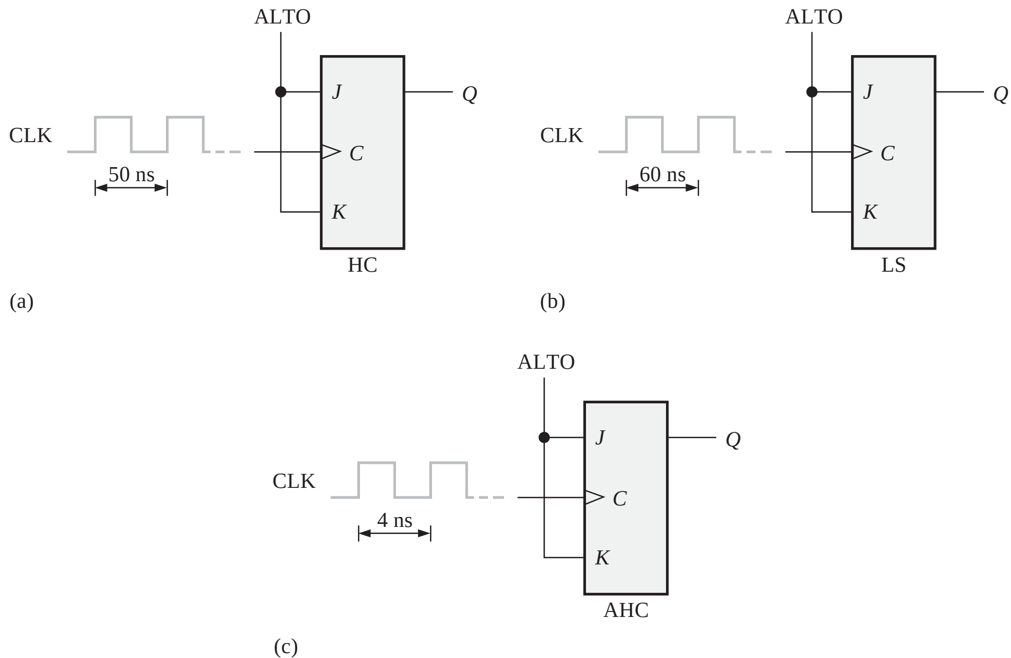


FIGURA 14.62

30. Elegir entre la tecnología ECL, la tecnología HCMOS o la apropiada serie TTL para cada uno de los siguientes requisitos:

(a) muy alta velocidad

(b) muy baja potencia

(c) mejor compromiso entre alta velocidad y baja potencia (producto velocidad-potencia)

RESPUESTAS

REVISIONES DE CADA SECCIÓN

SECCIÓN 14.1 Parámetros y características de operación básicas

1. V_{IH} : tensión de entrada a nivel ALTO. V_{IL} : tensión de entrada a nivel BAJO. V_{OH} : tensión de salida a nivel ALTO. V_{OL} : tensión de salida a nivel BAJO.

2. Es mejor un valor muy alto de margen de ruido.
3. La puerta *B* puede funcionar a una frecuencia mayor.
4. Una carga excesiva puede reducir el margen de ruido de una puerta.

SECCIÓN 14.2 Circuitos CMOS

1. Se utilizan transistores MOSFET en la lógica CMOS.
2. Un circuito con salida complementaria está formado por un MOSFET de canal-*n* y otro de canal-*p*.
3. Una descarga electrostática puede dañar los dispositivos CMOS.

SECCIÓN 14.3 Circuitos TTL

1. Falso. El BJT *npn* no conduce.
2. Un BJT en el estado de saturación (*on*) es un interruptor cerrado y en el estado de no conducción (*off*) es un interruptor abierto.
3. *Totem-pole* y colector abierto son dos tipos de salida TTL.
4. La lógica de tres estados proporciona un estado de alta impedancia, en el que la salida está desconectada del resto del circuito.

SECCIÓN 14.4 Consideraciones prácticas sobre el uso de TTL

1. Actúa como sumidero de corriente en el estado BAJO.
2. La corriente de fuente es menor que la corriente de sumidero porque una carga TTL es como un diodo polarizado en inversa en el estado ALTO.
3. Los transistores *totem-pole* no pueden soportar la corriente cuando una salida trata de adoptar un nivel ALTO y la otra un nivel BAJO.
4. La función AND-cableada debe utilizar salidas en colector abierto.
5. El excitador de lámpara debe tener una salida en colector abierto.
6. Falso, una entrada TTL no conectada generalmente actúa como un nivel ALTO.

SECCIÓN 14.5 Comparación de las prestaciones CMOS y TTL

1. BiCMOS utiliza transistores bipolares para los circuitos de entrada y salida y CMOS para los circuitos internos.
2. CMOS tiene una disipación de potencia más baja que la tecnología bipolar.

SECCIÓN 14.6 Circuitos ECL

1. ECL es más rápida que TTL
2. ECL consume más y tiene menor margen de ruido que TTL.

SECCIÓN 14.7 PMOS, NMOS y E^2CMOS

1. NMOS y PMOS son de alta densidad
2. La puerta flotante es el mecanismo para almacenar la carga en una celda E^2CMOS.

PROBLEMAS RELACIONADOS

14.1 CMOS

14.2 10,75 μW

14.3 $I_{T(fuente)} = 5\ (20\ \mu A) = 100\ \mu A$

$I_{T(sumidero)} = 5\ (-0{,}4\ mA) = -2{,}0\ mA$

14.4 *Fan-out* = 20

14.5 $X = (\overline{AB})(\overline{CD})(\overline{EF})(\overline{GH}) = (\bar{A} + \bar{B})(\bar{C} + \bar{D})(\bar{E} + \bar{F})(\bar{G} + \bar{H})$

14.6. *Véase* la Figura 14.63.

14.7 $R_L = 97\ \Omega$

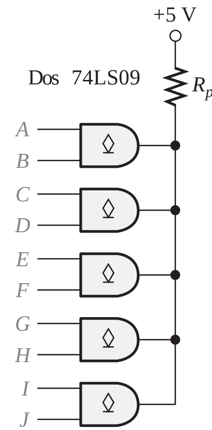


FIGURA 14.63

AUTOTEST

1. (b) **2.** (c) **3.** (c) **4.** (c) **5.** (e)
6. (a) **7.** (c) **8.** (d) **9.** (a) **10.** (c)

A

CONVERSIONES

Decimal	BCD (8421)	Octal	Binario	Decimal	BCD (8421)	Octal	Binario	Decimal	BCD (8421)	Octal	Binario
0	0000	0	0	26	00100110	32	11010	52	01010010	64	110100
1	0001	1	1	27	00100111	33	11011	53	01010011	65	110101
2	0010	2	10	28	00101000	34	11100	54	01010100	66	110110
3	0011	3	11	29	00101001	35	11101	55	01010101	67	110111
4	0100	4	100	30	00110000	36	11110	56	01010110	70	111000
5	0101	5	101	31	00110001	37	11111	57	01010111	71	111001
6	0110	6	110	32	00110010	40	100000	58	01011000	72	111010
7	0111	7	111	33	00110011	41	100001	59	01011001	73	111011
8	1000	10	1000	34	00110100	42	100010	60	01100000	74	111100
9	1001	11	1001	35	00110101	43	100011	61	01100001	75	111101
10	00010000	12	1010	36	00110110	44	100100	62	01100010	76	111110
11	00010001	13	1011	37	00110111	45	100101	63	01100011	77	111111
12	00010010	14	1100	38	00111000	46	100110	64	01100100	100	1000000
13	00010011	15	1101	39	00111001	47	100111	65	01100101	101	1000001
14	00010100	16	1110	40	01000000	50	101000	66	01100110	102	1000010
15	00010101	17	1111	41	01000001	51	101001	67	01100111	103	1000011
16	00010110	20	10000	42	01000010	52	101010	68	01101000	104	1000100
17	00010111	21	10001	43	01000011	53	101011	69	01101001	105	1000101
18	00011000	22	10010	44	01000100	54	101100	70	01110000	106	1000110
19	00011001	23	10011	45	01000101	55	101101	71	01110001	107	1000111
20	00100000	24	10100	46	01000110	56	101110	72	01110010	110	1001000
21	00100001	25	10101	47	01000111	57	101111	73	01110011	111	1001001
22	00100010	26	10110	48	01001000	60	110000	74	01110100	112	1001010
23	00100011	27	10111	49	01001001	61	110001	75	01110101	113	1001011
24	00100100	30	11000	50	01010000	62	110010	76	01110110	114	1001100
25	00100101	31	11001	51	01010001	63	110011	77	01110111	115	1001101

Decimal	BCD (8421)	Octal	Binario	Decimal	BCD (8421)	Octal	Binario	Decimal	BCD (8421)	Octal	Binario
78	01111000	116	1001110	86	10000110	126	1010110	93	10010011	135	1011101
79	01111001	117	1001111	87	10000111	127	1010111	94	10010100	136	1011110
80	10000000	120	1010000	88	10001000	130	1011000	95	10010101	137	1011111
81	10000001	121	1010001	89	10001001	131	1011001	96	10010110	140	1100000
82	10000010	122	1010010	90	10010000	132	1011010	97	10010111	141	1100001
83	10000011	123	1010011	91	10010001	133	1011011	98	10011000	142	1100010
84	10000100	124	1010100	92	10010010	134	1011100	99	10011001	143	1100011
85	10000101	125	1010101								

Potencias de dos

2^n	n	2^{-n}
1	0	1.0
2	1	0.5
4	2	0.25
8	3	0.125
16	4	0.062 5
32	5	0.031 25
64	6	0.015 625
128	7	0.007 812 5
256	8	0.003 906 25
512	9	0.001 953 125
1 024	10	0.000 976 562 5
2 048	11	0.000 488 281 25
4 096	12	0.000 244 140 625
8 192	13	0.000 122 070 312 5
16 384	14	0.000 061 035 156 25
32 768	15	0.000 030 517 578 125
65 536	16	0.000 015 258 789 062 5
131 072	17	0.000 007 629 394 531 25
262 144	18	0.000 003 814 697 265 625
524 288	19	0.000 001 907 348 632 812 5
1 048 576	20	0.000 000 953 674 316 406 25
2 097 152	21	0.000 000 476 837 158 203 125
4 194 304	22	0.000 000 238 418 579 101 562 5
8 388 608	23	0.000 000 119 209 289 550 781 25
16 777 216	24	0.000 000 059 604 644 775 390 625
33 554 432	25	0.000 000 029 802 322 387 695 312 5
67 108 864	26	0.000 000 014 901 161 193 847 656 25
134 217 728	27	0.000 000 007 450 580 596 923 828 125
268 435 456	28	0.000 000 003 725 290 298 461 914 062 5
536 870 912	29	0.000 000 001 862 645 149 230 957 031 25
1 073 741 824	30	0.000 000 000 931 322 574 615 478 515 625
2 147 483 648	31	0.000 000 000 465 661 287 307 739 257 812 5
4 294 967 296	32	0.000 000 000 232 830 643 653 869 628 906 25
8 589 934 592	33	0.000 000 000 116 415 321 826 934 814 453 125
17 179 869 184	34	0.000 000 000 058 207 660 913 467 407 226 562 5
34 359 738 368	35	0.000 000 000 029 103 830 456 733 703 613 281 25
68 719 476 736	36	0.000 000 000 014 551 915 228 366 851 806 640 625
137 438 953 472	37	0.000 000 000 007 275 957 614 183 425 903 320 312 5
274 877 906 944	38	0.000 000 000 003 637 978 807 091 712 951 660 156 25
549 755 813 888	39	0.000 000 000 001 818 989 403 545 856 475 830 078 125
1 099 511 627 776	40	0.000 000 000 000 909 494 701 772 928 237 915 039 062 5
2 199 023 255 552	41	0.000 000 000 000 454 747 350 886 464 118 957 519 531 25
4 398 046 511 104	42	0.000 000 000 000 227 373 675 443 232 059 478 759 765 625
8 796 093 022 208	43	0.000 000 000 000 113 686 837 721 616 029 739 379 882 812 5
17 592 186 044 416	44	0.000 000 000 000 056 843 418 860 808 014 869 689 941 406 25
35 184 372 088 832	45	0.000 000 000 000 028 421 709 430 404 007 434 844 970 703 125
70 368 744 177 664	46	0.000 000 000 000 014 210 854 715 202 003 717 422 485 351 562 5
140 737 488 355 328	47	0.000 000 000 000 007 105 427 357 601 001 858 711 242 675 781 25
281 474 976 710 656	48	0.000 000 000 000 003 552 713 678 800 500 929 355 621 337 890 625
562 949 953 421 312	49	0.000 000 000 000 001 776 356 839 400 250 464 677 810 668 945 312 5
1 125 899 906 842 624	50	0.000 000 000 000 000 888 178 419 700 125 232 338 905 334 472 656 25
2 251 799 813 685 248	51	0.000 000 000 000 000 444 089 209 850 062 616 169 452 667 236 328 125
4 503 599 627 370 496	52	0.000 000 000 000 000 222 044 604 925 031 308 084 726 333 618 164 062 5
9 007 199 254 740 992	53	0.000 000 000 000 000 111 022 302 462 515 654 042 363 166 809 082 031 25
18 014 398 509 481 984	54	0.000 000 000 000 000 055 511 151 231 257 827 021 181 583 404 541 015 625
36 028 797 018 963 968	55	0.000 000 000 000 000 027 755 575 615 628 913 510 590 791 702 270 507 812 5
72 057 594 037 927 936	56	0.000 000 000 000 000 013 877 787 807 814 456 755 295 395 851 135 253 906 25
144 115 188 075 855 872	57	0.000 000 000 000 000 006 938 893 903 907 228 377 647 697 925 567 626 953 125
288 230 376 151 711 744	58	0.000 000 000 000 000 003 469 446 951 953 614 188 823 848 962 783 813 476 562 5
576 460 752 303 423 488	59	0.000 000 000 000 000 001 734 723 475 976 807 094 411 924 481 391 906 738 281 25
1 152 921 504 606 846 976	60	0.000 000 000 000 000 000 867 361 737 988 403 547 205 962 240 695 953 369 140 625
2 305 843 009 213 693 952	61	0.000 000 000 000 000 000 433 680 868 994 201 773 602 981 120 347 976 684 570 312 5
4 611 686 018 427 387 904	62	0.000 000 000 000 000 000 216 840 434 497 100 886 801 490 560 173 988 342 285 156 25
9 223 372 036 854 775 808	63	0.000 000 000 000 000 000 108 420 217 248 550 443 400 745 280 086 994 171 142 578 125
18 446 744 073 709 551 616	64	0.000 000 000 000 000 000 054 210 108 624 275 221 700 372 640 043 497 085 571 289 062 5
36 893 488 147 419 103 232	65	0.000 000 000 000 000 000 027 105 054 312 137 610 850 186 320 021 748 542 785 644 531 25
73 786 976 294 838 206 464	66	0.000 000 000 000 000 000 013 552 527 156 088 805 425 093 160 010 874 271 392 822 265 625
147 573 952 589 676 412 928	67	0.000 000 000 000 000 000 006 776 263 578 034 402 712 546 580 005 437 135 696 411 132 812 5
295 147 905 179 352 825 856	68	0.000 000 000 000 000 000 003 388 131 789 017 201 356 273 290 002 718 567 848 205 566 406 25
590 295 810 358 705 651 712	69	0.000 000 000 000 000 000 001 694 065 894 508 600 678 136 645 001 359 283 924 102 783 203 125
1 180 591 620 717 411 303 424	70	0.000 000 000 000 000 000 000 847 032 947 254 300 339 068 322 500 679 641 962 051 391 601 562 5
2 361 183 241 434 822 606 848	71	0.000 000 000 000 000 000 000 423 516 473 627 150 169 534 161 250 339 820 981 025 695 800 781 25
4 722 366 482 869 645 213 696	72	0.000 000 000 000 000 000 000 211 758 236 813 575 084 767 080 625 169 910 490 512 847 900 390 625

B

INTERFAZ DE LAS LUCES DE LOS SEMÁFOROS

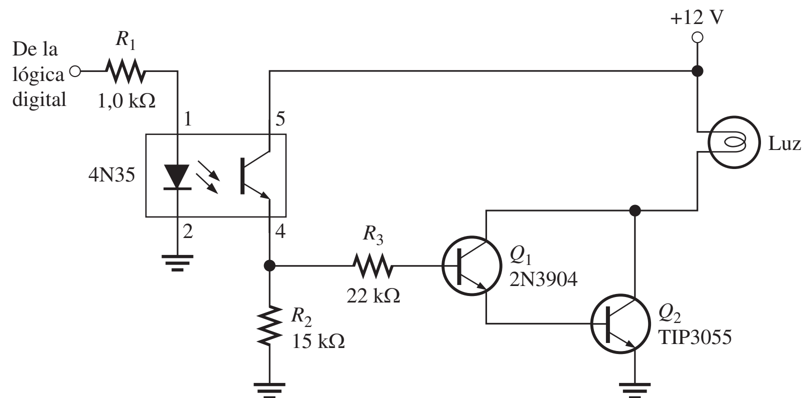


FIGURA B.1 Circuito de interfaz utilizado con las luces de los semáforos. Un circuito excita una luz.



Tarjeta de desarrollo que contiene el CPLD programado y una tarjeta de interfaz que controla en el laboratorio los semáforos del modelo. Cortesía de Dave Buchla.

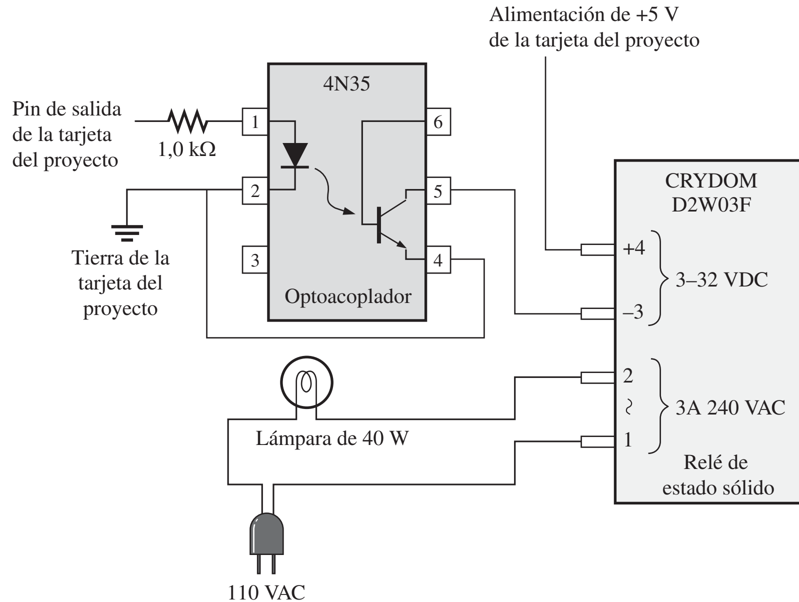
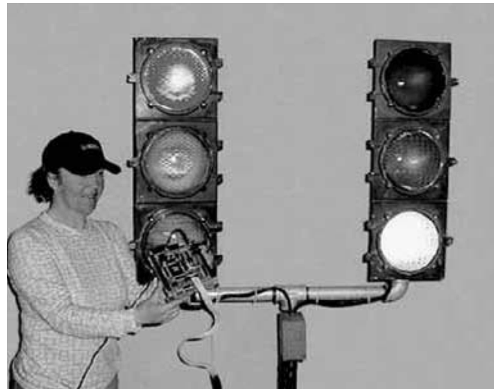


FIGURA B.2 Circuito de interfaz utilizado con las luces de los semáforos. Un circuito excita una luz.



Una estudiante sostiene en el laboratorio una tarjeta de desarrollo que contiene un CPLD programado que controla las luces de los semáforos reales. Los circuitos de interfaz se encuentran en el interior de una caja metálica montada en el soporte de los semáforos. Cortesía de Doug Joksch.

RESPUESTAS A LOS PROBLEMAS IMPARES

CAPÍTULO 1

1. Los datos digitales pueden transmitirse y almacenarse de forma más eficiente y fiable.

3. **(a)** 11010001 **(b)** 000101010

5. **(a)** 550 ns **(b)** 600 ns
(c) 2,7 μs **(d)** 10 V

7. 250 Hz

9. 50%

11. 8 μs; 1 μs

13. Puerta AND

15. **(a)** sumador **(b)** multiplicador
(c) multiplexor **(d)** comparador

17. 01010000

19. Los terminales de un encapsulado DIP se insertan en los taladros de una tarjeta de circuito impreso. Los terminales de un dispositivo de montaje superficial se conectan en los *pad* superficiales.

21. ABEL, CUPL

23. **(a)** Introducción del diseño: es el paso en el flujo de un diseño de lógica programable en el que se introduce una descripción del circuito bien en forma de esquemático (gráfico) o en forma de texto utilizando un lenguaje HDL.
(b) Simulación: es el paso en el flujo de diseño en el que se simula el diseño introducido basándose en las formas de onda de entrada definidas.
(c) Compilación: un proceso del programa que controla el proceso de flujo de diseño y traduce el código fuente del diseño en código objeto que será probado y descargado.
(d) Descarga: el proceso en el que se transfiere el diseño software a hardware.

25. 7 V

27. Una colección de circuitos interconectados para realizar una función específica.

29. Introducir un valor nuevo a través del teclado.

CAPÍTULO 2

1. **(a)** 1 **(b)** 100 **(c)** 100.000

3. **(a)** 400; 70; 1
(b) 9000; 300; 50; 6
(c) 100.000; 20.000; 5000; 0; 0; 0

5. **(a)** 3 **(b)** 4 **(c)** 7 **(d)** 8
(e) 9 **(f)** 12 **(g)** 11 **(h)** 15

7. **(a)** 51,75 **(b)** 42,25 **(c)** 65,875 **(d)** 120,625
(e) 92,65625 **(f)** 113,0625 **(g)** 90,625 **(h)** 127,96875

9. **(a)** 5 bits **(b)** 6 bits **(c)** 6 bits **(d)** 7 bits
(e) 7 bits **(f)** 7 bits **(g)** 8 bits **(h)** 8 bits

11. **(a)** 1010 **(b)** 10001 **(c)** 11000
(d) 110000 **(e)** 111101 **(f)** 1011101
(g) 1111101 **(h)** 10111010

13. **(a)** 1111 **(b)** 10101 **(c)** 11100
(d) 100010 **(e)** 101000 **(f)** 111011
(g) 1000001 **(h)** 1001001

15. **(a)** 100 **(b)** 100 **(c)** 1000
(d) 1101 **(e)** 1110 **(f)** 11000

17. **(a)** 1001 **(b)** 1000 **(c)** 100011
(d) 110110 **(e)** 10101001 **(f)** 10110110

19. **(a)** 010 **(b)** 001 **(c)** 0101
(d) 00101000 **(e)** 0001010 **(f)** 11110

21. **(a)** 00011101 **(b)** 11010101 **(c)** 01100100 **(d)** 11111011

23. **(a)** 00001100 **(b)** 10111100 **(c)** 01100101 **(d)** 10000011

25. **(a)** −102 **(b)** +116 **(c)** −64

27. **(a)** 0 10001101 11110000101011000000000
(b) 1 10001010 11000001100000000000000

29. **(a)** 00110000 **(b)** 00011101 **(c)** 11101011 **(d)** 100111110

31. **(a)** 11000101 **(b)** 11000000

33. 100111001010

35. **(a)** 00111000 **(b)** 01011001 **(c)** 101000010100
(d) 010111001000 **(e)** 0100000100000000
(f) 1111101100010111 **(g)** 1000101010011101

37. **(a)** 35 **(b)** 146 **(c)** 26 **(d)** 141
(e) 243 **(f)** 235 **(g)** 1474 **(h)** 1792

39. **(a)** 60_{16} **(b)** $10B_{16}$ **(c)** $1BA_{16}$

41. **(a)** 10 **(b)** 23 **(c)** 46 **(d)** 52 **(e)** 67
(f) 367 **(g)** 115 **(h)** 532 **(i)** 4085

43. **(a)** 001011 **(b)** 101111
(c) 001000001 **(d)** 011010001
(e) 101100000 **(f)** 100110101011
(g) 001011010111001 **(h)** 100101110000000
(i) 0010000000010001011

45. **(a)** 00010000 **(b)** 00010011 **(c)** 00011000
(d) 00100001 **(e)** 00100101 **(f)** 00110110
(g) 01000100 **(h)** 01010111 **(i)** 01101001
(j) 10011000 **(k)** 000100100101
(l) 000101010110

47. **(a)** 000100000100 **(b)** 000100101000
(c) 000100110010 **(d)** 000101010000
(e) 000110000110 **(f)** 001000010000
(g) 001101011001 **(h)** 010101000111
(i) 0001000001010001

49. **(a)** 80 **(b)** 237 **(c)** 346 **(d)** 421 **(e)** 754
(f) 800 **(g)** 978 **(h)** 1683 **(i)** 9018 **(j)** 6667

51. **(a)** 00010100 **(b)** 00010010
(c) 00010111 **(d)** 00010110
(e) 01010010 **(f)** 000100001001
(g) 000110010101 **(h)** 0001001001101001

53. El código Gray hace que sólo cambie un bit cada vez cuando pasa de un número de la secuencia al siguiente.

55. **a)** 1100 **(b)** 00011 **(c)** 10000011110

57. **(a)** CAN **(b)** J **(c)** = **(d)** # **(e)** > **(f)** B

59. 48 65 6C 6C 6F 2E 20 48 6F 77 20 61 72 65 20 79 6F 75 3F

61. **(b)** Es incorrecto.

63. **(a)** 110100100 **(b)** 000001001 **(c)** 111111110

65. 001010001

67. **(a)** 110100010 **(b)** 100000101

CAPÍTULO 3

1. *Véase* la Figura P.1.

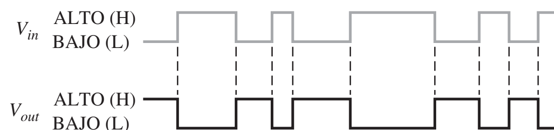


FIGURA P.1

3. *Véase* la Figura P.2.

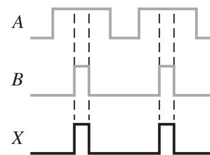


FIGURA P.2

5. *Véase* la Figura P.3.

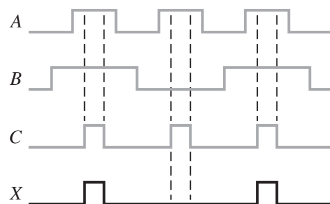


FIGURA P.3

7. *Véase* la Figura P.4.

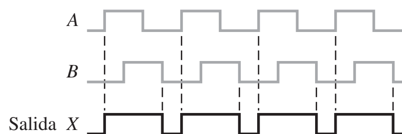


FIGURA P.4

9. *Véase* la Figura P.5.

11. *Véase* la Figura P.6.

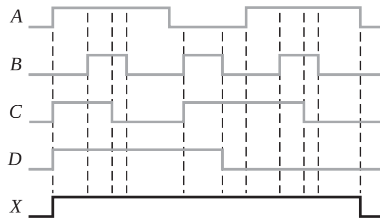


FIGURA P.5

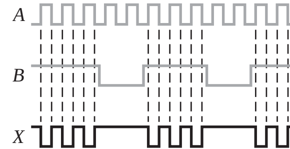


FIGURA P.6

13. *Véase* la Figura P.7.

15. *Véase* la Figura P.8.

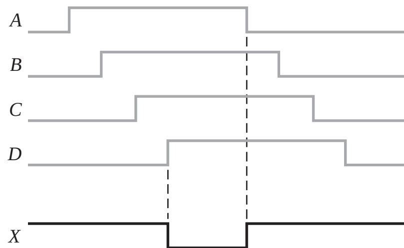


FIGURA P.7

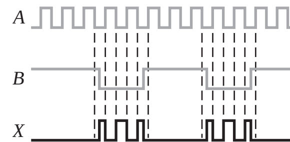


FIGURA P.8

17. *Véase* la Figura P.9.

19. XOR $= A\overline{B} + \overline{A}B$; OR $= A + B$

21. *Véase* la Figura P10.

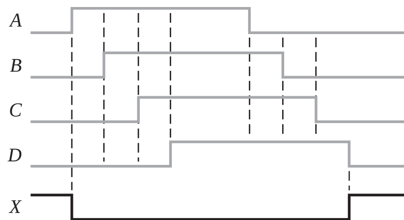


FIGURA P.9

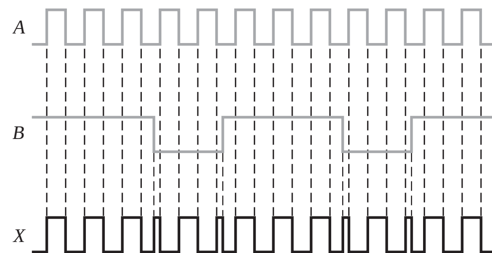


FIGURA P.10

23. $X_1 = \overline{A}B,\ X_2 = \overline{A}\overline{B},\ X_3 = A\overline{B}$

25. CMOS

27. $t_{PLH} = 4{,}3$ ns; $t_{PHL} = 10{,}5$ ns.

29. 20 mW

31. Las puertas de los apartados (b), (c) y (e) fallan.

33. **(a)** Salida defectuosa (está en circuito abierto o se mantiene a nivel BAJO)

(b) La entrada del pin 4 o la salida del pin 6 están internamente en circuito abierto.

35. La entrada de cinturón de seguridad a la puerta AND está en circuito abierto.

37. *Véase* la Figura P11.

39. Añadir un inversor a la línea de entrada de habilitación de la puerta AND.

41. *Véase* la Figura P12.

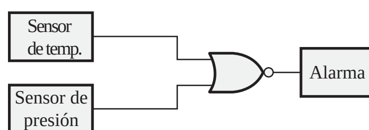


FIGURA P.11

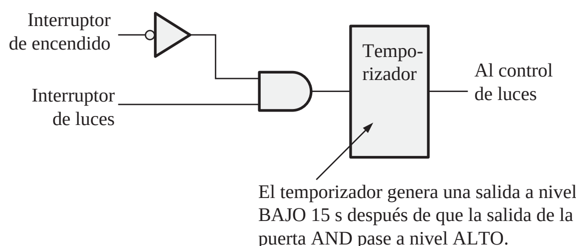


FIGURA P.12

43. Las entradas ahora son activas a nivel BAJO. Cambiar las puertas OR por puertas NAND (negativa-OR) y añadir dos inversores.

CAPÍTULO 4

1. $X = A + B + C + D$

3. $X = \bar{A} + \bar{B} + \bar{C}$

5. **(a)** $AB = 1$ cuando $A = 1, B = 1$
(b) $A\bar{B}C = 1$ cuando $A = 1, B = 0, C = 1$
(c) $A + B = 0$ cuando $A = 0, B = 0$
(d) $\bar{A} + B + \bar{C} = 0$ cuando $A = 1, B = 0, C = 1$
(e) $\bar{A} + \bar{B} + C = 0$ cuando $A = 1, B = 1, C = 0$
(f) $\bar{A} + B = 0$ cuando $A = 1, B = 0$
(g) $A\bar{B}\bar{C} = 1$ cuando $A = 1, B = 0, C = 0$

7. **(a)** Conmutativa **(b)** Conmutativa **(c)** Distributiva

9. **(a)** $\bar{A}B$ **(b)** $A + \bar{B}$
(c) $\bar{A}\bar{B}\bar{C}$ **(d)** $\bar{A} + \bar{B} + \bar{C}$
(e) $\bar{A} + \bar{B}\bar{C}$ **(f)** $\bar{A} + \bar{B} + \bar{C} + \bar{D}$
(g) $(\bar{A} + \bar{B})(\bar{C} + \bar{D})$ **(h)** $\bar{A}B + C\bar{D}$

11. **(a)** $(\bar{A} + \bar{B} + \bar{C})(\bar{E} + \bar{F} + \bar{G})(\bar{H} + \bar{I} + \bar{J})(\bar{K} + \bar{L} + \bar{M})$
(b) $\bar{A}B\bar{C} + BC$
(c) $\bar{A}\bar{B}\bar{C}\bar{D}\bar{E}\bar{F}\bar{G}\bar{H}$

13. **(a)** $X = ABCD$ **(b)** $X = AB + C$
(c) $X = \overline{\bar{A}B}$ **(d)** $X = (A + B)C$

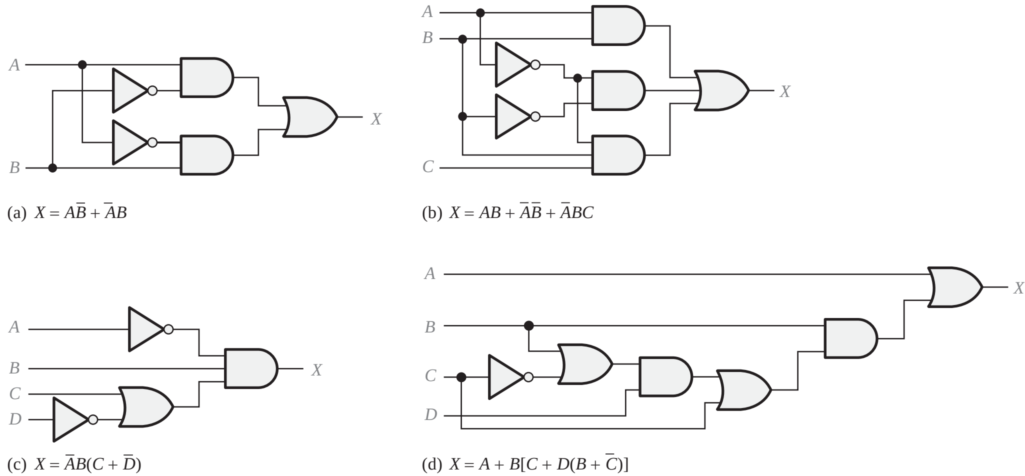
15. *Véase* la Figura P.13.

17. **(a)** A **(b)** AB **(c)** C **(d)** A **(e)** $\bar{A}C + \bar{B}C$

19. **(a)** $BD + BE + \bar{D}F$ **(b)** $\bar{A}\bar{B}C + \bar{A}\bar{B}D$
(c) B **(d)** $AB + CD$ **(e)** ABC

21. **(a)** $A\bar{B} + AC + BC$ **(b)** $AC + \bar{B}C$ **(c)** $AB + AC$

23. Dominio: A, B, C
Suma de productos estándar: $A\bar{B}C + A\bar{B}\bar{C} + ABC + \bar{A}BC$
Dominio: A, B, C



(a) $X = A\overline{B} + \overline{A}B$

(b) $X = AB + \overline{A}\overline{B} + \overline{A}BC$

(c) $X = \overline{A}B(C + \overline{D})$

(d) $X = A + B[C + D(B + \overline{C})]$

FIGURA P.13

Suma de productos estándar: $ABC + A\overline{B}C + \overline{A}\overline{B}C$

Dominio: A, B, C

Suma de productos estándar: $ABC + AB\overline{C} + A\overline{B}C$

25. **(a)** 101 + 100 + 111 + 011 **(b)** 111 + 101 + 001 **(c)** 111 + 110 + 101

27. **(a)** $(A+B+C)(A+B+\overline{C})(A+\overline{B}+C)(\overline{A}+\overline{B}+C)$

(b) $(A+B+C)(A+\overline{B}+C)(A+\overline{B}+\overline{C})(\overline{A}+B+C)(\overline{A}+\overline{B}+C)$

(c) $(A+B+C)(A+B+\overline{C})(A+\overline{B}+C)(A+\overline{B}+\overline{C})(\overline{A}+B+C)$

29. **(a)** *Véase* la Tabla P.1. **(b)** *Véase* la Tabla P.2.

A	*B*	*C*	*X*
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

TABLA P.1

X	*Y*	*Z*	*Q*
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

TABLA P.2

31. **(a)** *Véase* la Tabla P.3. **(b)** *Véase* la Tabla P.4.

33. **(a)** *Véase* la Tabla P.5. **(b)** *Véase* la Tabla P.6.

35. *Véase* la Figura P.14.

37. *Véase* la Figura P.15.

39. **(a)** No puede simplificarse. **(b)** AC **(c)** $\overline{D}\overline{F} + E\overline{F}$

A	*B*	*C*	*X*
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

TABLA P.3

W	*X*	*Y*	*Z*	*Q*
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

TABLA P.4

A	*B*	*C*	*X*
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

TABLA P.5

A	*B*	*C*	*D*	*X*
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

TABLA P.6

AB \ C	0	1
00	000	001
01	010	011
11	110	111
10	100	101

FIGURA P.14

AB \ C	0	1
00	$\bar{A}\bar{B}\bar{C}$	$\bar{A}\bar{B}C$
01	$\bar{A}B\bar{C}$	$\bar{A}BC$
11	$AB\bar{C}$	ABC
10	$A\bar{B}\bar{C}$	$A\bar{B}C$

FIGURA P.15

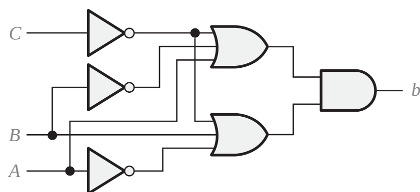
41. **(a)** $AB + AC$ **(b)** $A + BC$
(c) $B\bar{C}D + A\bar{C}D + BC\bar{D} + AC\bar{D}$ **(d)** $A\bar{B} + CD$

43. $\bar{B} + C$

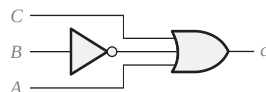
45. $\bar{A}\bar{B}\bar{C}D + C\bar{D} + BC + A\bar{D}$

47. **(a)** $(A + \bar{B} + C + \bar{D})(\bar{A} + B + \bar{C} + D)(\bar{A} + \bar{B} + \bar{C} + \bar{D})$
(b) $(W + \bar{Z})(W + X)(\bar{Y} + \bar{Z})(X + \bar{Y})$

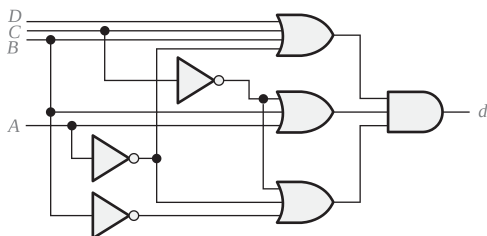
49. $(A + C + D)(A + \bar{B} + C)(\bar{A} + B + \bar{D})(B + \bar{C} + \bar{D})(\bar{A} + \bar{B} + \bar{C} + D)$



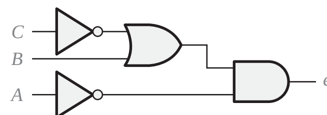
segmento $b = (\bar{C} + B + \bar{A})(\bar{C} + \bar{B} + A)$



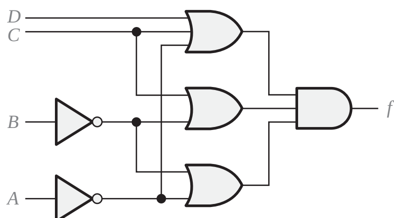
segmento $c = C + \bar{B} + A$



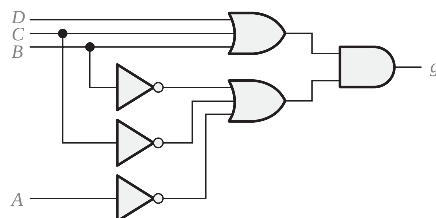
segmento $d = (D + C + B + \bar{A})(\bar{C} + B + A)(\bar{C} + \bar{B} + \bar{A})$



segmento $e = \bar{A}(\bar{C} + B)$



segmento $f = (D + C + \bar{A})(C + \bar{B})(\bar{B} + \bar{A})$



segmento $g = (D + C + B)(\bar{C} + \bar{B} + \bar{A})$

FIGURA P.16

51. $X = \overline{A}\,\overline{B}\,\overline{C}\,\overline{D}\,\overline{E} + \overline{A}BC\overline{D}E + AB\overline{C}\,\overline{D}\,\overline{E} + \overline{A}\,\overline{B}DE + \overline{A}BD\overline{E} + \overline{B}\,\overline{C}DE + AB\overline{C}D$

53.
```
entity AND_OR is
        port (A, B, C, D, E, F, G, H, I: in bit; X: out bit);
    end entity AND_OR;
    architecture Logic of AND_OR is
    begin
    X <= (A and B and C) or (D and E and F) or
    (G and H and I);
         end architecture Logic;
```

55. LED. Los LED emiten luz, los LCD no.

57. Un inversor y seis puertas menos.

59. Añadir un inversor a la salida de la puerta OR de cada uno de los circuitos lógicos de los segmentos.

61. *Véase* la Figura P.16.

CAPÍTULO 5

1. *Véase* la Figura P.17.

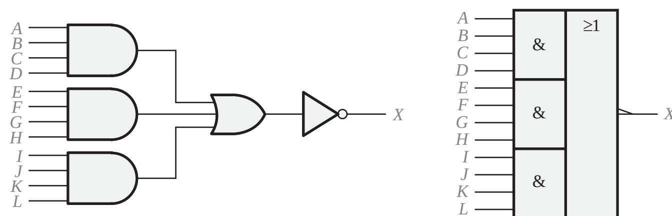


FIGURA P.17

3.
(a) $X = ABB$
(b) $X = AB + B$
(c) $X = \overline{A} + B$
(d) $X = (A + B) + AB$
(e) $X = \overline{\overline{A}BC}$
(f) $X = (A + B)(\overline{B} + C)$

5. (a)

A	*B*	*X*
0	0	0
0	1	0
1	0	0
1	1	1

5. (b)

A	*B*	*X*
0	0	0
0	1	1
1	0	0
1	1	1

5. (c)

A	*B*	*X*
0	0	1
0	1	1
1	0	0
1	1	1

5. (d)

A	*B*	*X*
0	0	0
0	1	1
1	0	1
1	1	1

5. **(e)**

A	*B*	*C*	*X*
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

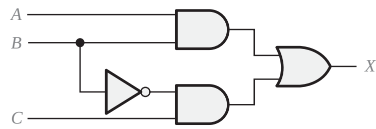
5. **(f)**

A	*B*	*C*	*X*
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

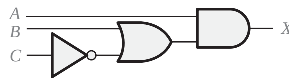
7. $X = \overline{A\overline{B} + \overline{A}B} = (A + \overline{B})(\overline{A} + B)$

9. *Véase* la Figura P.18.

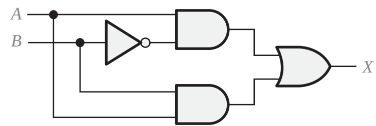
11. *Véase* la Figura P.19.



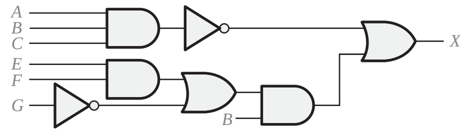
(a) $X = AB + \overline{B}C$



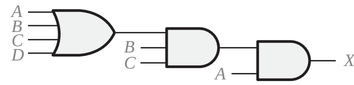
(b) $X = A(B + \overline{C})$



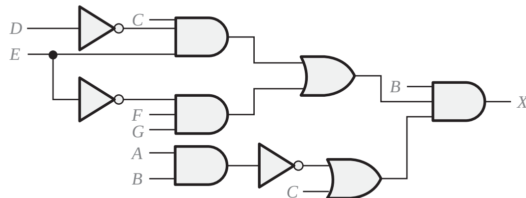
(c) $X = A\overline{B} + AB$



(d) $X = \overline{ABC} + B(EF + \overline{G})$



(e) $X = A[BC(A + B + C + D)]$



(f) $X = B(C\overline{D}E + \overline{E}FG)\,(\overline{AB} + C)$

FIGURA P.18

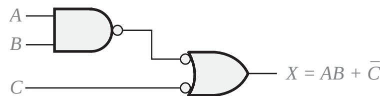


FIGURA P.19

13. $X = AB$

15. **(a)** No puede simplificarse.

(b) No puede simplificarse.

(c) $X = A$

(d) $X = \overline{A} + \overline{B} + \overline{C} + EF + \overline{G}$

(e) $X = ABC$

(f) $X = BC\overline{D}E + \overline{A}B\overline{E}FG + BC\overline{E}FG$

17. **(a)** $X = AC + AD + BC + BD$

(b) $X = \overline{A}CD + \overline{B}CD$

(c) $X = ABD + CD + E$

(d) $X = \overline{A} + B + D$

(e) $X = ABD + \overline{C}D + \overline{E}$

(f) $X = \overline{A}\overline{C} + \overline{A}\overline{D} + \overline{B}\overline{C} + \overline{B}\overline{D} + \overline{E}\overline{G} + \overline{E}\overline{H} + \overline{F}\overline{G} + \overline{F}\overline{H}$

19. *Véase* la Figura P.20.

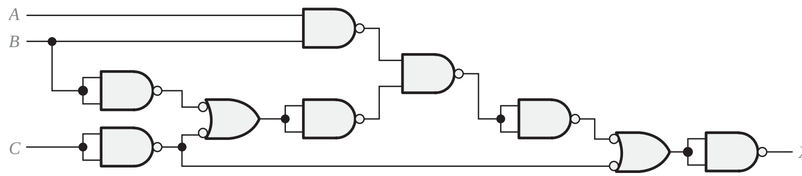


FIGURA P.20

21. *Véase* la Figura P.21.

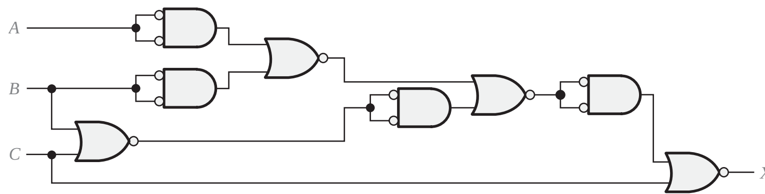
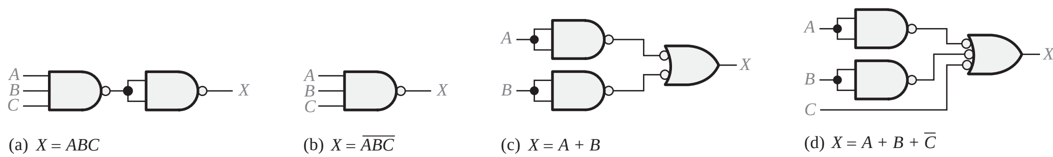
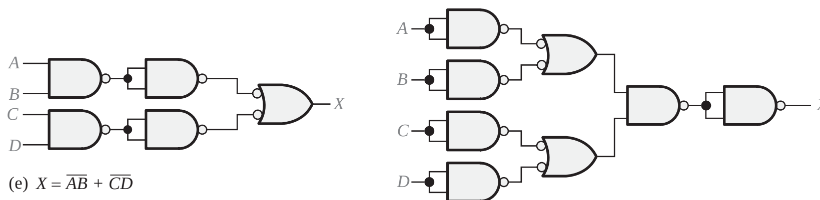


FIGURA P.21

23. *Véase* la Figura P.22.

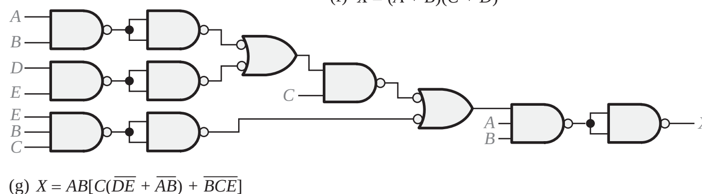


(a) $X = ABC$ (b) $X = \overline{ABC}$ (c) $X = A + B$ (d) $X = A + B + \overline{C}$



(e) $X = \overline{AB} + \overline{CD}$

(f) $X = (A + B)(C + D)$



(g) $X = AB[C(\overline{DE} + \overline{AB}) + \overline{BCE}]$

FIGURA P.22

25. *Véase* la Figura P.23.

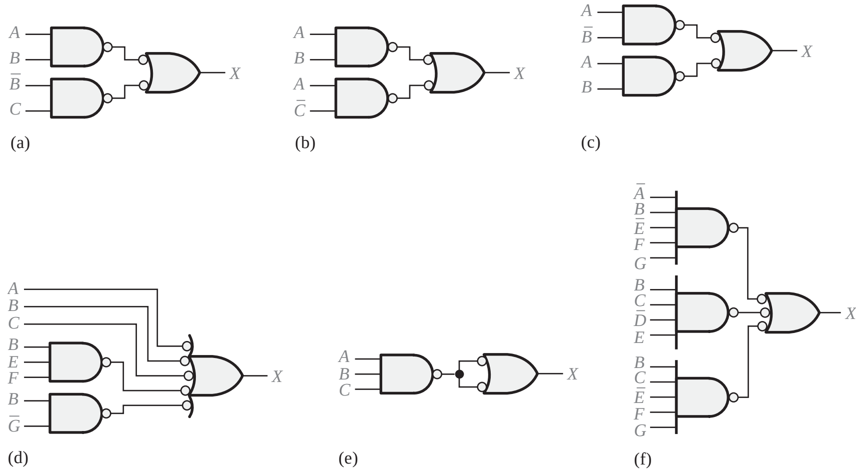


FIGURA P.23

27. $X = A + \overline{B}$; *véase* la Figura P.24.

29. $X = A\overline{B}\overline{C}$; *véase* la Figura P.25.

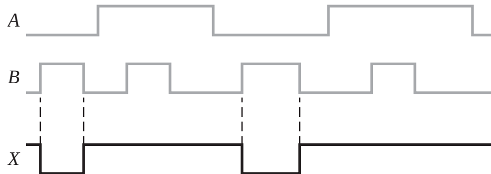


FIGURA P.24

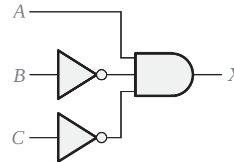


FIGURA P.25

31. La anchura del impulso de salida es mayor que el mínimo especificado.

33. **(e)**

```
entity Circuit 5_52e is
    port (A, B, C: in bit; X: out bit);
  end entity Circuit5_52e;
  architecture LogicFunction of Circuit5_52e is
  begin
     X < = (not A and B) or B or (B and not C) or
      (not A and not C) or (B and not C) or not C;
  end architecture LogicFunction;
```

(f)

```
entity Circuit5_52f is
    port (A, B, C: in bit; X: out bit);
  end entity Circuit5_52f;
  architecture LogicFunction of Circuit5_52f is
  begin
    X <= (A or B) and (not B or C);
  end architecture LogicFunction;
```

35. Numerar las puertas de arriba a abajo y de izquierda a derecha G1, G2, G3, etc. Reetiquetar las entradas IN1, IN2, IN3, etc. y la salida OUT.

```
entity Circuit5_53f is
  port (IN1, IN2, IN3, IN4, IN5, IN6, IN7, IN8: in bit;
OUT: out bit);
end entity Circuit_53f
architecture LogicFunction of Circuit5_53f is
component NAND_gate is
  port (A, B: in bit; X: out bit);
end component NAND_gate;
  signal G1OUT, G2OUT, G3OUT, G4OUT, G5OUT, G6OUT: bit;
begin
  G1: NAND_gate port map (A => IN1, B => IN2, X => G1OUT);
  G2: NAND_gate port map (A => IN3, B => IN4, X => G2OUT);
  G3: NAND_gate port map (A => IN5, B => IN6, X => G3OUT);
  G4: NAND_gate port map (A => IN7, B => IN8, X => G4OUT);
  G5: NAND_gate port map (A => G1OUT, B => G2OUT, X => G5OUT);
  G6: NAND_gate port map (A => G3OUT, B => G4OUT, X => G6OUT);
  G7: NAND_gate port map (A => G5OUT, B => G6OUT, X => OUT);
end architecture LogicFunction;
```

37.

```
--Método del flujo de datos
entity Fig5_64 is
  port (A, B, C, D, E: in bit; X: out bit);
end entity Fig5_64;
architecture DataFlow of Fig5_64 is
begin
  X <= (A and B and C) or (D and not E);
end architecture DataFlow;
--Método estructural
entity Fig5_64 is
  port (IN1, IN2, IN3, IN4, IN5: in bit; OUT: out bit);
end entity Fig5_64;
architecture Structure of Fig5_64 is
component AND_gate is
  port (A, B: in bit; X: out bit);
end component AND_gate;
component OR_gate is
 port (A, B: in bit; X: out bit);
end component OR_gate;
component Inverter is
  port (A: in bit; X: out bit);
end component Inverter;
```

```
  signal G1OUT, G2OUT, G3OUT, INVOUT: bit;
begin
  G1: AND_gate port map (A => IN1, B => IN2, X => G1OUT);
  G2: AND_gate port map (A => G1OUT, B =_> IN3, X => G2OUT);
  INV: Inverter port map (A => IN5, X => INVOUT);
  G3: AND_gate port map (A => IN4, B => INVOUT, X => G3OUT);
  G4: OR_gate port map (A => G2OUT, B => G3OUT, X => OUT);
end architecture Structure;
```

39. *Véase* la Tabla P.7.

Entradas				Salida
A	*B*	*C*	*D*	*X*
0	0	0	0	0
1	0	0	0	0
0	1	0	0	0
1	1	0	0	0
0	0	1	0	0
1	0	1	0	0
0	1	1	0	0
1	1	1	0	0
0	0	0	1	0
1	0	0	1	0
0	1	0	1	0
1	1	0	1	1
0	0	1	1	0
1	0	1	1	1
0	1	1	1	1
1	1	1	1	1

TABLA P.7

41. Las puertas AND están numeradas de arriba a abajo como G1, G2, G3, G4. La puerta OR es G5 y los inversores son, de arriba a abajo, G6 y G7. Cambiar A_1, A_2, B_1, B_2 por IN1, IN2, IN3, IN4, respectivamente. Cambiar X por OUT.

```
entity Circuit5_62 is
  port (IN1, IN2, IN3, IN4: in bit; OUT: out bit);
end entity Circuit5_62;
architecture Logic of Circuit5_62 is
component AND_gate is
  port (A, B: in bit; X: out bit);
end component AND_gate;
component OR_gate is
  port (A, B, C, D: in bit; X: out bit);
end component OR_gate;
```

```
component Inverter is
  port (A: in bit; X: out bit);
end component Inverter;
  signal G1OUT, G2OUT, G3OUT, G4OUT, G5OUT, G6OUT, G7OUT: bit;
begin
  G1: AND_gate port map (A => IN1, B => IN2, X => G1OUT);
  G2: AND_gate port map (A => IN2, B => G6OUT, X => G2OUT);
  G3: AND_gate port map (A => G6OUT, B => G7OUT, X => G3OUT);
  G4: AND_gate port map (A => G7OUT, B => IN1, X => G4OUT);
  G5: OR_gate port map (A => G1OUT, B => G2OUT, C => G3OUT, D => G4OUT, X => OUT);
 G6: Inverter port map (A => IN3, X => G6OUT);
  G7: Inverter port map (A => IN4, X => G7OUT);
end architecture Logic;
```

43. $X = ABC + D\overline{E}$. Puesto que X es igual a la salida de G_3, G_1 o G_2 han fallado manteniendo su salida a nivel BAJO.

45. **(a)** *Véase* la Figura P.26.

47. **(a)** $X = E$. *Véase* la Figura P.27. **(b)** $X = E$ **(c)** $X = E$

49. *Véase* la Figura P.28.

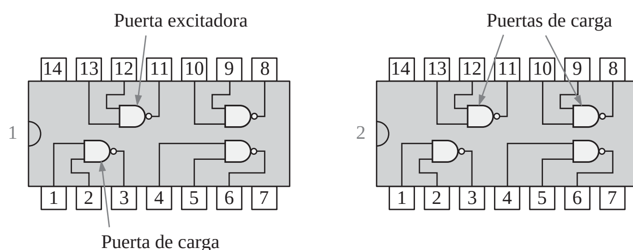


FIGURA P.26

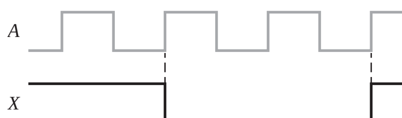


FIGURA P.27

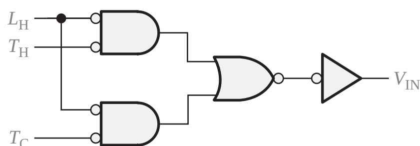
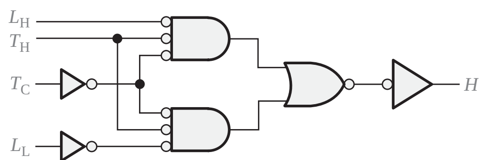


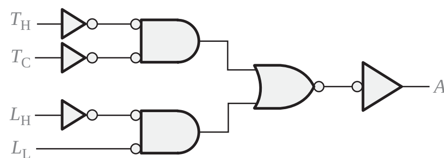
FIGURA P.28

51. *Véase* la Figura P.29.

53. X = lámpara encendida. A = interruptor frente a la puerta activado. B = interruptor detrás de la puerta activado. *Véase* la Figura P.30.



Lógica de la calefacción



Lógica de la alarma

FIGURA P.29



FIGURA P.30

55. *Véase* la Figura P.31. Los inversores (no mostrados) se usan para convertir cada nivel alto de pulsación de tecla a nivel BAJO.

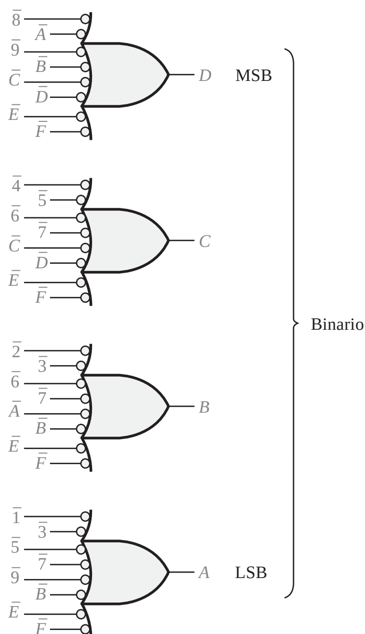


FIGURA P.31

CAPÍTULO 6

1. $A \oplus B = 0, \Sigma = 1, (A \oplus B)\, C_{in} = 0, AB = 1, C_{out} = 1$

$A \oplus B = 1, \Sigma = 0, (A \oplus B)\, C_{in} = 1, AB = 0, C_{out} = 1$

$A \oplus B = 1, \Sigma = 1, (A \oplus B)\, C_{in} = 0, AB = 0, C_{out} = 0$

3. **(a)** $\Sigma = 1, C_{out} = 0$; **(b)** $\Sigma = 1, C_{out} = 0$;
(c) $\Sigma = 0, C_{out} = 1$; **(d)** $\Sigma = 1, C_{out} = 1$

5. 11100

7. $\Sigma_1 = 0110$; $\Sigma_2 = 1011$; $\Sigma_3 = 0110$; $\Sigma_4 = 0001$; $\Sigma_5 = 1000$

9. 225 ns

11. $A = B$ es un nivel ALTO cuando $A_0 = B_0$ y $A_1 = B_1$. *Véase* la Figura P.32.

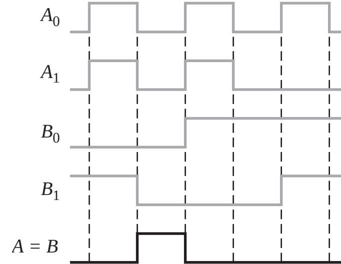


FIGURA P.32

13. **(a)** $A > B = 1$; $A = B = 0$; $A < B = 0$.

(b) $A < B = 1$; $A = B = 0$; $A > B = 0$.

(c) $A = B = 1$; $A < B = 0$; $A > B = 0$.

15. *Véase* la Figura P.33.

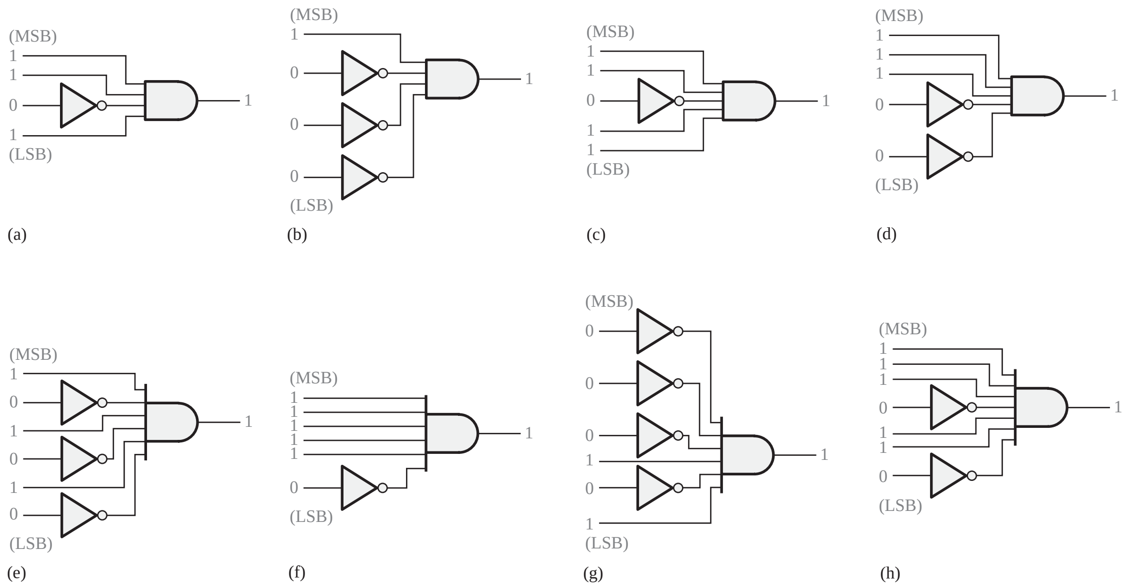


FIGURA P.33

17. $X = A_3A_2\bar{A}_1\bar{A}_0 + \bar{A}_3\bar{A}_2\bar{A}_1A_0 + A_3\bar{A}_2A_1$

19. *Véase* la Figura P.34.

21. $A_3A_2A_1A_0 = 1011$. Código BCD no válido.

23. **(a)** $2 = 0010 = 0010_2$ **(b)** $8 = 1000 = 1000_2$

(c) $13 = 00010011 = 1101_2$ **(d)** $26 = 00100110 = 11010_2$

(e) $33 = 00110011 = 100001_2$

25. **(a)** 1010000000 Gray → 1100000000 binario

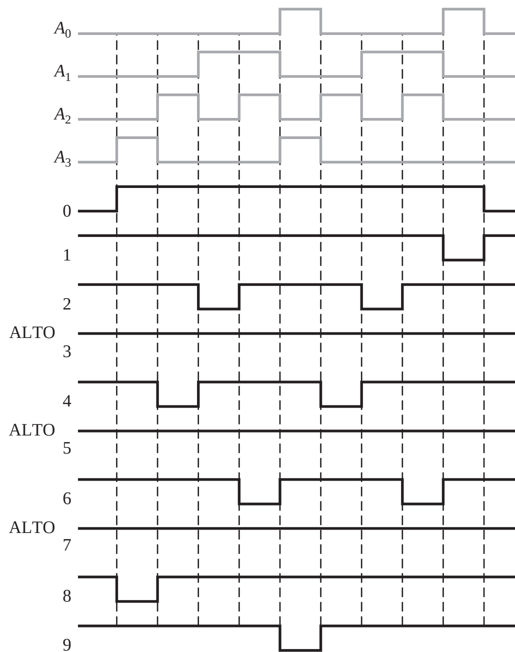


FIGURA P.34

(b) 0011001100 Gray → 0010001000 binario

(c) 1111000111 Gray → 1010000101 binario

(d) 0000000001 Gray → 0000000001 binario

Véase la Figura P.35.

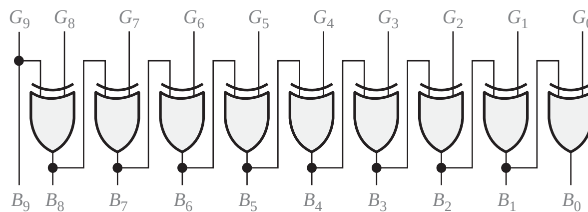


FIGURA P.35

27. *Véase* la Figura P.36.

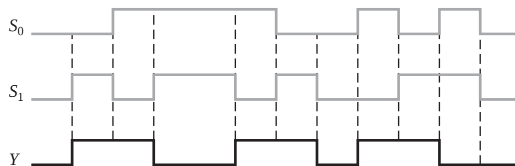


FIGURA P.36

29. *Véase* la Figura P.37.

31. *Véase* la Figura P.38.

33. **(a)** Correcto.

(b) el segmento *g* está fundido, la salida G en abierto.

(c) la salida del segmento *b* se mantiene a nivel BAJO.

35. **(a)** La entrada A_1 del sumador superior está en circuito abierto. Todos los valores binarios correspondientes al número BCD que tienen un valor de 0, 1, 4, 5, 8 o 9 se desplazan 2 unidades. El primer valor BCD para el que se detecta el error es 0000 0000.

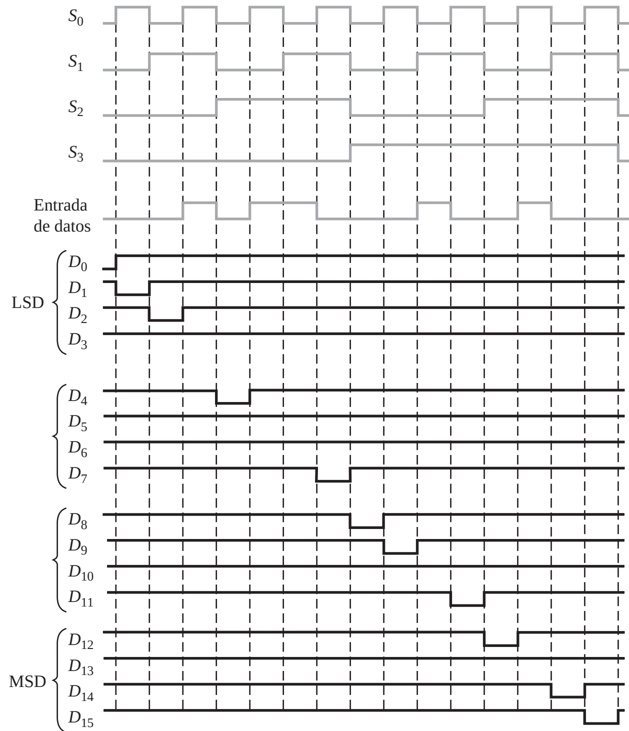


FIGURA P.37

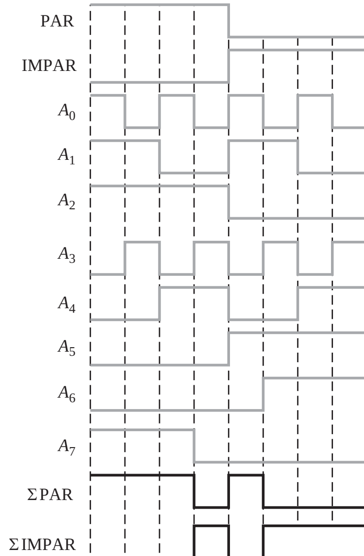


FIGURA P.38

(b) La salida de acarreo del sumador superior está en circuito abierto. Normalmente, no todos los valores que implican un acarreo de salida se desplazarán 32 unidades. El primer valor BCD para el que se detecta el error es 0000 0000.

(c) La salida Σ_4 del sumador superior está cortocircuitada a tierra. El mismo conjunto de valores anterior se reducirá en 16 unidades. El primer valor que indica esto es 0001 1000.

(d) La salida Σ_3 del sumador inferior está cortocircuitada a tierra: cada grupo de 16 valores, comenzando por 16, se reducirá en 16 unidades. El primer valor BCD que indica esto es 0001 0110.

37. **1.** Poner a nivel BAJO el pin 7 (habilitación).

2. Aplicar un nivel ALTO a D_0 y un nivel BAJO a D_1 hasta D_7.

3. Aplicar la secuencia binaria a las entradas de selección y comprobar Y e $\overline{Y}$ de acuerdo con la Tabla P.8.

4. Repetir la secuencia binaria de las entradas de selección para cada uno de los conjuntos de entradas de datos enumerados en la Tabla P.9. Un nivel ALTO en la salida Y sólo debería producirse para las combinaciones de las entradas de selección indicadas.

S_2	S_1	S_0	Y	$\overline{Y}$
0	0	0	1	0
0	0	1	0	1
0	1	0	0	1
0	1	1	0	1
1	0	0	0	1
1	0	1	0	1
1	1	0	0	1
1	1	1	0	1

TABLA P.8

D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	Y	$\overline{Y}$	S_2	S_1	S_0
L	H	L	L	L	L	L	L	1	0	0	0	1
L	L	H	L	L	L	L	L	1	0	0	1	0
L	L	L	H	L	L	L	L	1	0	0	1	1
L	L	L	L	H	L	L	L	1	0	1	0	0
L	L	L	L	L	H	L	L	1	0	1	0	1
L	L	L	L	L	L	H	L	1	0	1	1	0
L	L	L	L	L	L	L	H	1	0	1	1	1

L = nivel BAJO H = nivel ALTO

TABLA P.9

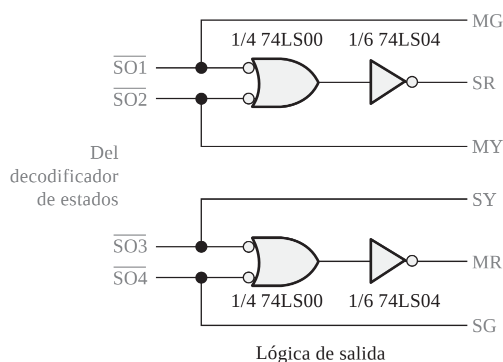


FIGURA P.39

39. Aplicar por turno un nivel alto a cada entrada de datos, D_0 hasta D_7, estando las demás entradas a nivel BAJO. Para cada nivel alto aplicado a una entrada de datos, seguir la secuencia de las ocho combinaciones binarias de las entradas de selección ($S_2S_1S_0$) y comprobar que se produce un nivel ALTO en la salida de datos correspondiente y que todas las restantes salidas de datos están a nivel BAJO.

41. *Véase* la Figura P.39.

43. $\Sigma = \bar{A}\bar{B}C_{\text{in}} + \bar{A}B\bar{C}_{\text{in}} + A\bar{B}\bar{C}_{\text{in}} + ABC_{\text{in}}$

$C_{\text{out}} = \bar{A}BC_{\text{in}} + A\bar{B}C_{\text{in}} + AB\bar{C}_{\text{in}} + ABC_{\text{in}}$

Véase la Figura P.40.

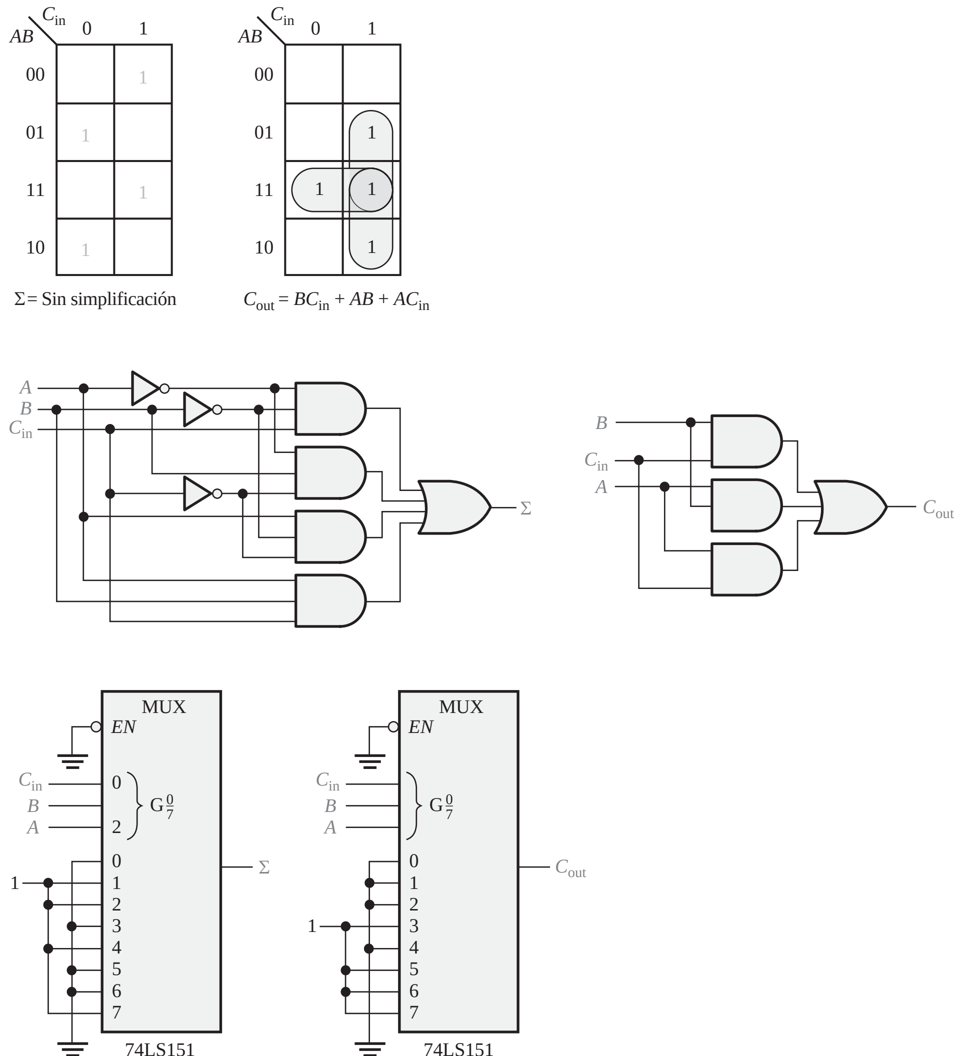


FIGURA P.40

45. *Véase* el diagrama de bloques del Figura P.41.

47. *Véase* la Figura P.42.

49. *Véase* la Figura P.43.

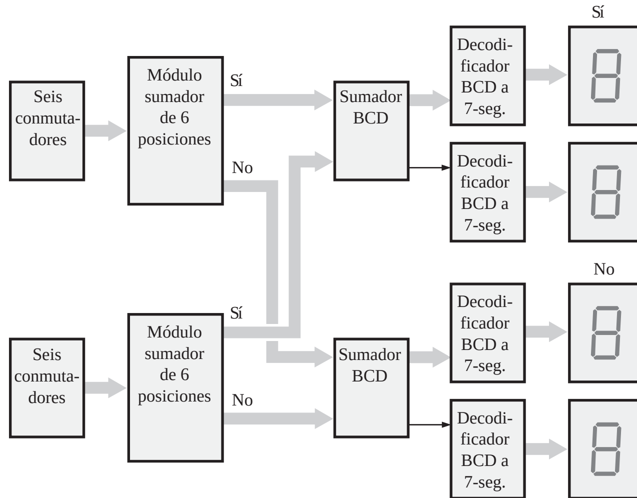


FIGURA P.41

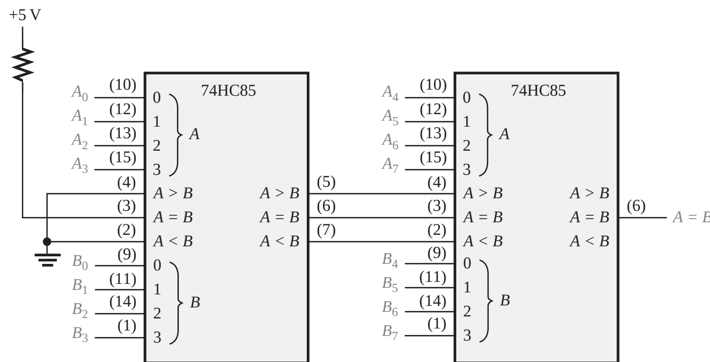


FIGURA P.42

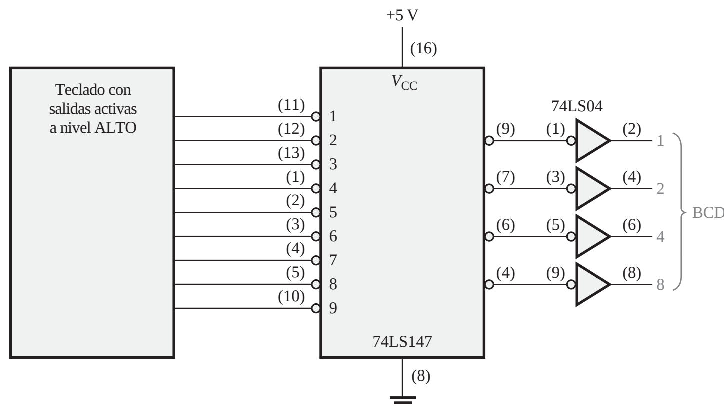


FIGURA P.43

CAPÍTULO 7

1. *Véase* la Figura P.44.

3. *Véase* la Figura P.45.

5. *Véase* la Figura P.46.

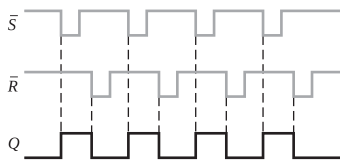


FIGURA P.44

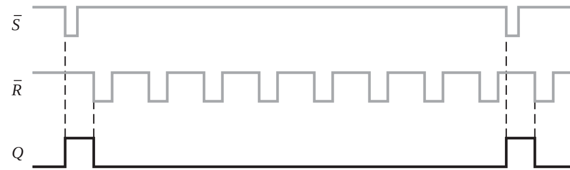


FIGURA P.45

7. *Véase* la Figura P.47.

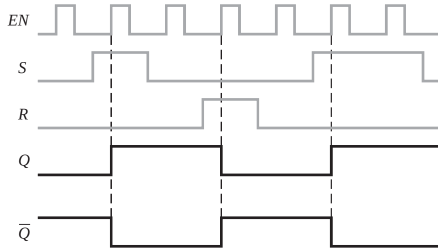


FIGURA P.46

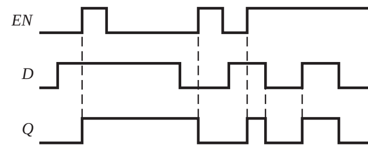


FIGURA P.47

9. *Véase* la Figura P.48.

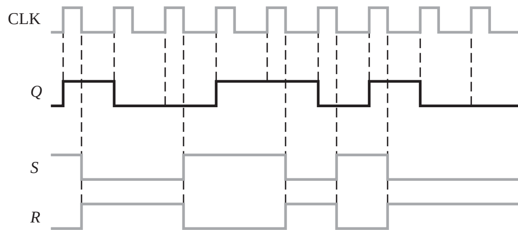


FIGURA P.48

11. *Véase* la Figura P.49.

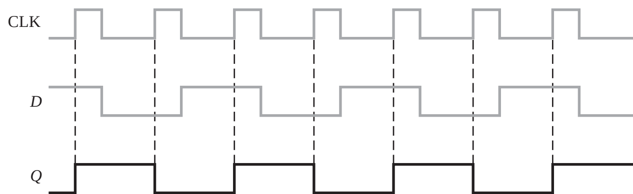


FIGURA P.49

13. *Véase* la Figura P.50.

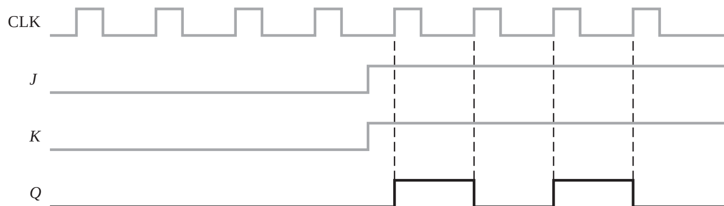


FIGURA P.50

15. *Véase* la Figura P.51.

17. *Véase* la Figura P.52.

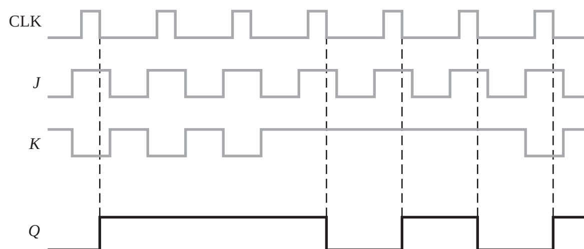


FIGURA P.51

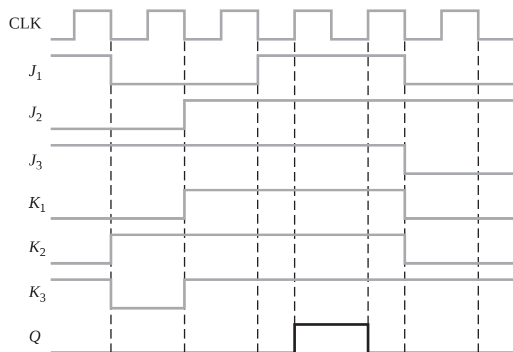


FIGURA P.52

19. Corriente continua y tensión de alimentación continua.

21. 14,9 MHz

23. 150 mA, 750 mW

25. Divisor por 2. *Véase* la Figura P.53.

27. 4,62 μs

29. $C_1 = 1\ \mu$F, $R_1 = 227$ kΩ (utilice 220 kΩ). *Véase* la Figura P.54.

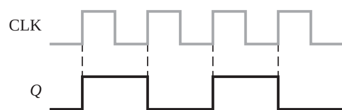


FIGURA P.53

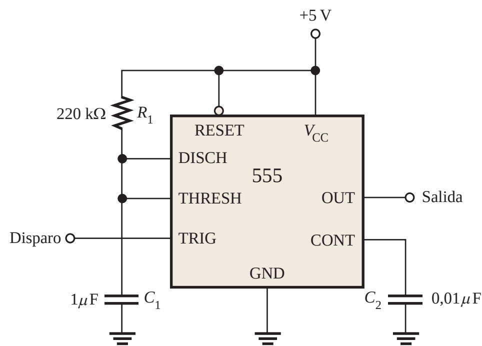


FIGURA P.54

31. $R_1 = 18$ kΩ. $R_2 = 9{,}1$ kΩ.

33. La conexión del pin 6 al pin 10 y la conexión de tierra están invertidos en el prototipo de la tarjeta.

35. $\overline{CLR}$ está cortocircuita a tierra.

37. *Véase* la Figura P.55. Los retardos no se indican.

39. *Véase* la Figura P.56.

4 s: $C_1 = 1\ \mu$F, $R_1 = 3{,}63$ MΩ (utilizar 3,9 MΩ)

25 s: $C_1 = 2,2\ \mu F$, $R_1 = 10,3 M\Omega$ (utilizar 10 MΩ)

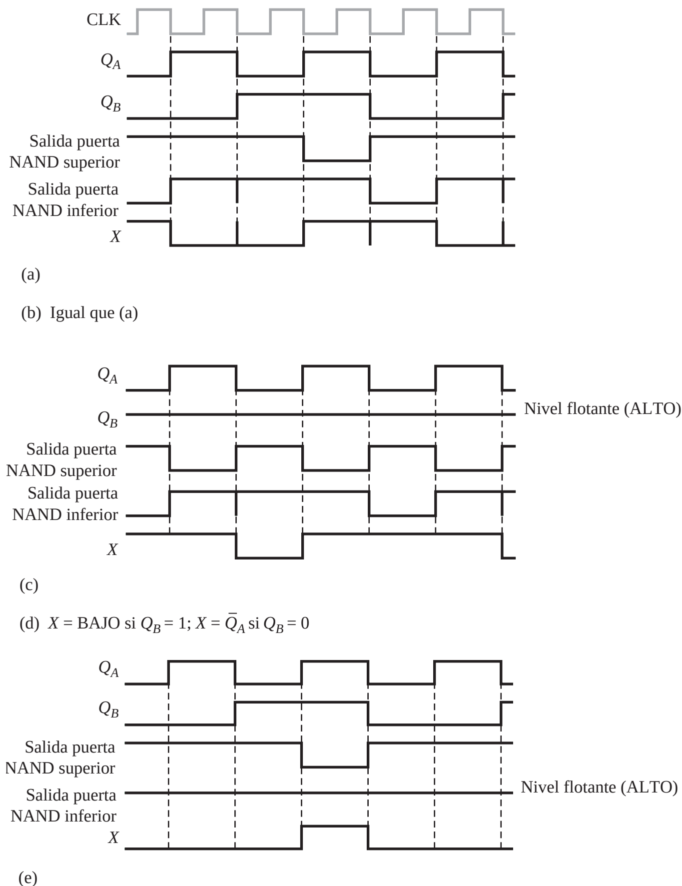


FIGURA P.55

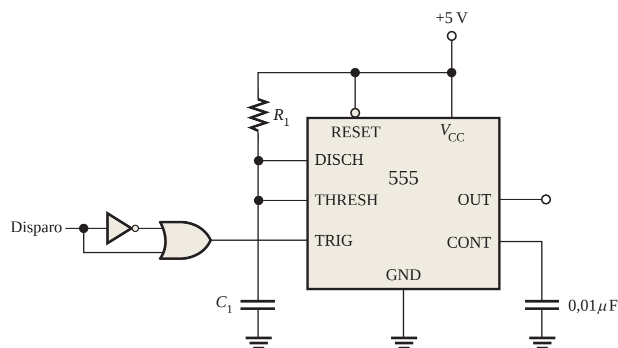


FIGURA P.56

41. *Véase* la Figura P.57.

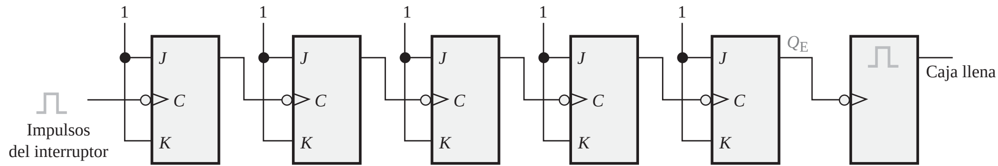


FIGURA P.57

CAPÍTULO 8

1. *Véase* la Figura P.58.

3. El retardo en el caso peor es de 24 ns. Se produce cuando todos los flip-flops cambian del estado 011 al 100, o del 111 al 000.

5. 8 ns.

7. Inicialmente, cada flip-flop está en estado de RESET.

En CLK1:

$J_0 = K_0 = 1$, por tanto Q_0 se pone a 1.
$J_1 = K_1 = 0$, por tanto Q_1 se mantiene a 0.
$J_2 = K_2 = 0$, por tanto Q_2 se mantiene a 0.
$J_3 = K_3 = 0$, por tanto Q_3 se mantiene a 0.

En CLK2:

$J_0 = K_0 = 1$, por tanto Q_0 se pone a 1.
$J_1 = K_1 = 1$, por tanto Q_1 se pone a 1.
$J_2 = K_2 = 0$, por tanto Q_2 se mantiene a 0.
$J_3 = K_3 = 0$, por tanto Q_3 se mantiene a 0.

En CLK3:

$J_0 = K_0 = 1$, por tanto Q_0 se pone a 1.
$J_1 = K_1 = 0$, por tanto Q_1 se mantiene a 0.
$J_2 = K_2 = 0$, por tanto Q_2 se mantiene a 0.
$J_3 = K_3 = 0$, por tanto Q_3 se mantiene a 0.

Siguiendo este proceso para los siete impulsos de reloj se verá que el contador progresa a través de una secuencia BCD.

9. *Véase* la Figura P.59.

11. *Véase* la Figura P.60.

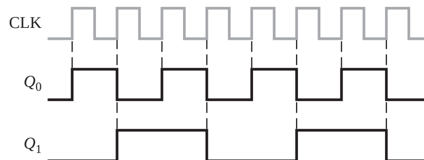


FIGURA P.58

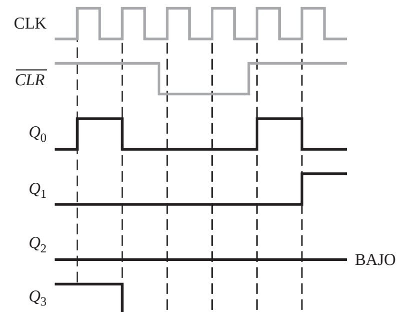


FIGURA P.59

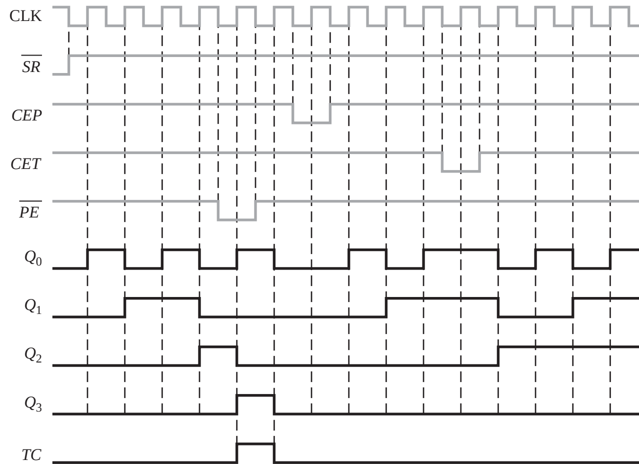


FIGURA P.60

13. *Véase* la Figura P.61.

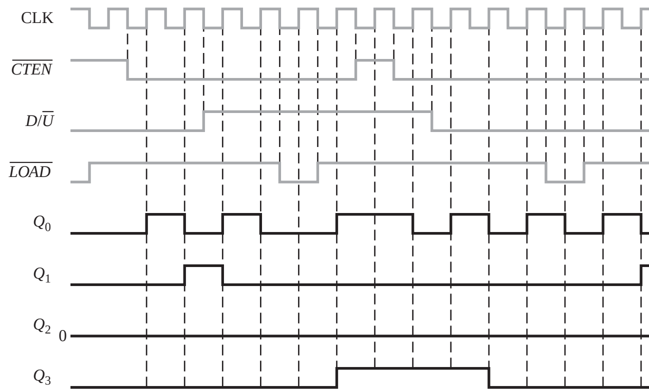


FIGURA P.61

15. La secuencia es 0000, 1111, 1110, 1101, 1010, 0101. El contador "se bloquea" en los estados 1010 y 0101 y alterna entre ellos.

17. *Véase* la Figura P.62.

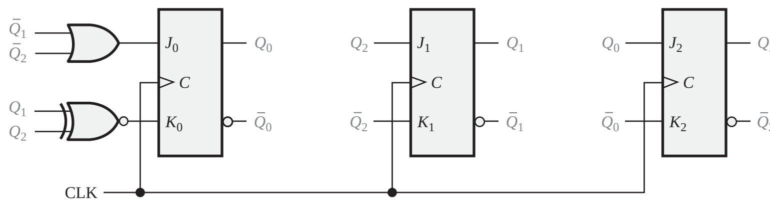


FIGURA P.62

19. *Véase* la Figura P.63.

21. *Véase* la Figura P.64 para la división por 10.000. Añadir un contador DIV10 más para crear un divisor por 100.000.

23. *Véase* la Figura P.65.

25. CLK2, salida 0; CLK4, salidas 2, 0; CLK6, salida 4; CLK8, salidas 6, 4, 0; CLK10, salida 8; CLK12, salidas 10, 8; CLK14, salida 12; CLK16, salidas 14, 12, 8.

27. Se produce un *glitch* en una puerta AND en la transición de 111 a 000. Se elimina aplicando la operación AND a $\overline{\text{CLK}}$ y las salidas del contador (*strobe*) o usando código Gray.

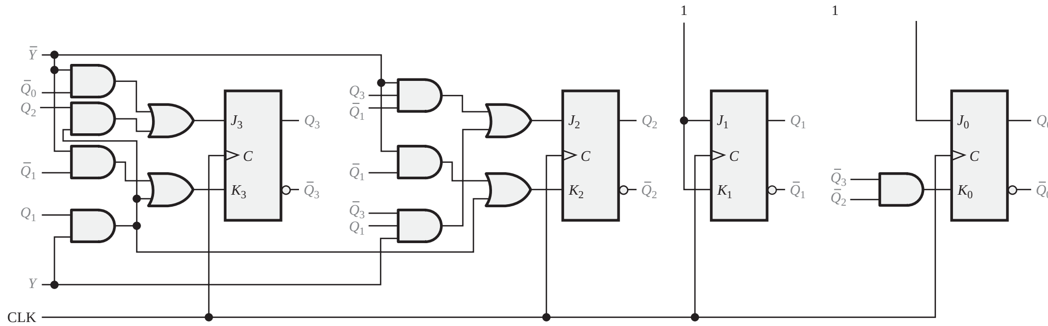


FIGURA P.63

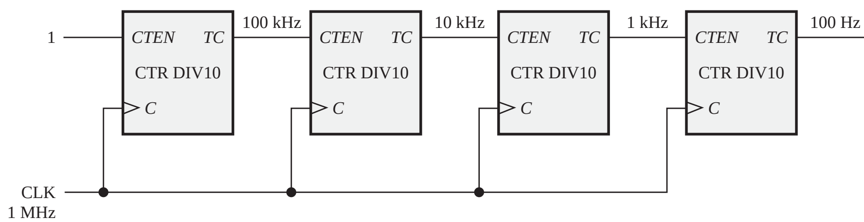


FIGURA P.64

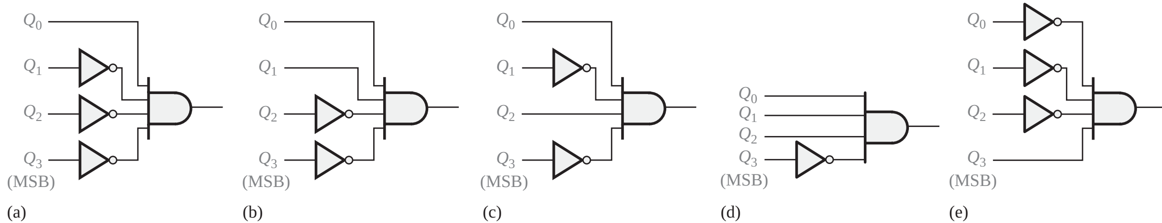


FIGURA P.65

29. Decenas de horas: 0001
Unidades de horas: 0010
Decenas de minutos: 0000
Unidades de minutos: 0001
Decenas de segundos: 0000
Unidades de segundos: 0010

31. 64

33. **(a)** Q_0 y Q_1 no cambiarán de su estado inicial.
(b) Funcionamiento normal excepto Q_0 flotante.
(c) La forma de onda Q_0 es normal; Q_1 permanece en el estado inicial.
(d) Funcionamiento normal.
(e) El contador no cambiará de su estado inicial.

35. La entrada *K* de FF1 debe conectarse a tierra en lugar de la entrada *J*. Comprobar un error de conexión.

37. La entrada Q_0 y la puerta AND están en circuito abierto y actúan como un nivel ALTO.

39. *Véase* la Tabla P.10.

Etapa	Abierto	Cuenta cargada	f_{OUT}
1	0	63C1	250,006 Hz
1	1	63C2	250,012 Hz
1	2	63C4	250,025 Hz
1	3	63C8	250,050 Hz
2	0	63D0	250,100 Hz
2	1	63E0	250,200 Hz
2	2	63C0	250 Hz
2	3	63C0	250 Hz
3	0	63C0	250 Hz
3	1	63C0	250 Hz
3	2	67C0	256,568 Hz
3	3	6BC0	263,491 Hz
4	0	73C0	278,520 Hz
4	1	63C0	250 Hz
4	2	63C0	250 Hz
4	3	E3C0	1,383 kHz

TABLA P.10

41. La puerta 6 de decodificación interpreta la cuenta 4 como 6 (0110) y borra el contador poniéndolo a 0 (realmente 0010, puesto que Q_1 está en circuito abierto). La secuencia aparente de la parte de las unidades del contador es 0010, 0011, 0010, 0011, 0110.

43. *Véase* la Figura P.66.

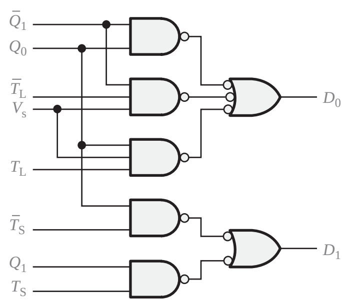


FIGURA P.66

45. Aumentar 2,4 veces la constante de tiempo R_{EXT} C_{EXT} del monoestable de 25 segundos.

47. *Véase* la Figura P.67.

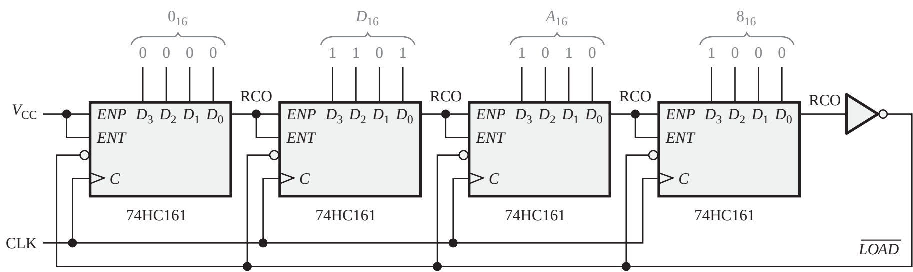


FIGURA P.67

49. *Véase* la Figura P.68.

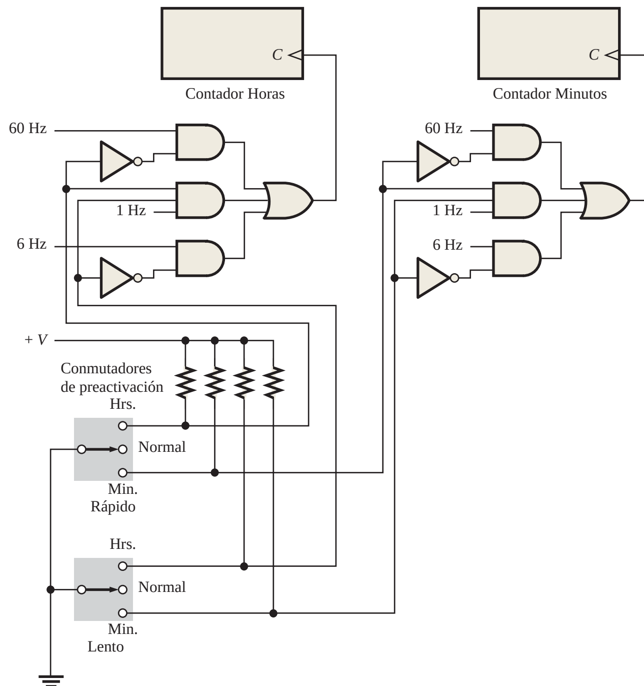


FIGURA P.68

51. *Véase* la Figura P.69.

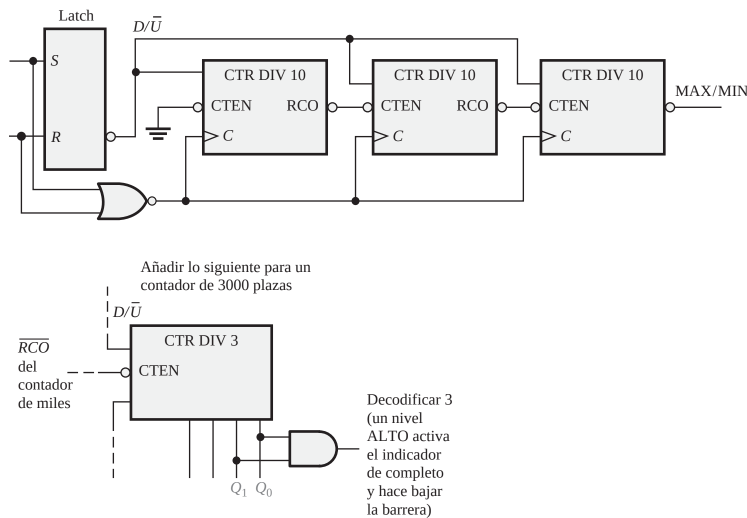


FIGURA P.69

53. *Véase* la Figura P.70.

CAPÍTULO 9

1. Los registros de desplazamiento almacenan datos binarios.

3. *Véase* la Figura P.71.

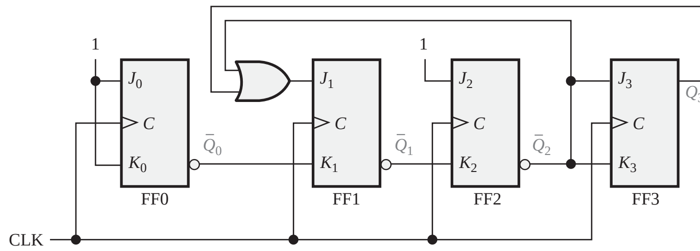


FIGURA P.70

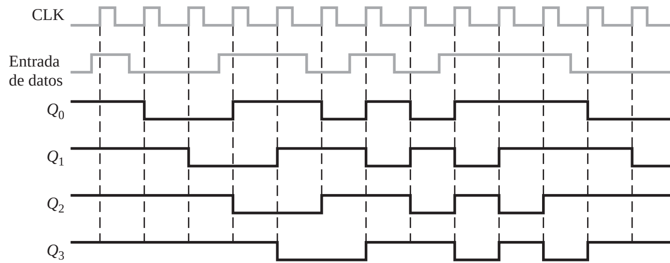


FIGURA P.71

5. Inicialmente: 101001111000
CLK1: 010100111100
CLK2: 001010011110
CLK3: 000101001111
CLK4: 000010100111
CLK5: 100001010011
CLK6: 110000101001
CLK7: 111000010100
CLK8: 011100001010
CLK9: 001110000101
CLK10: 000111000010
CLK11: 100011100001
CLK12: 110001110000

7. *Véase* la Figura P.72.

9. *Véase* la Figura P.73.

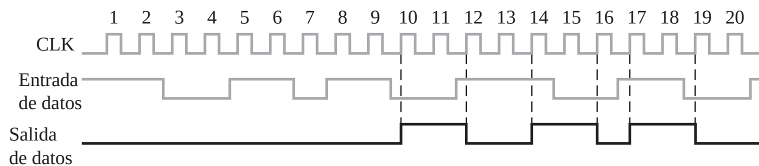


FIGURA P.72

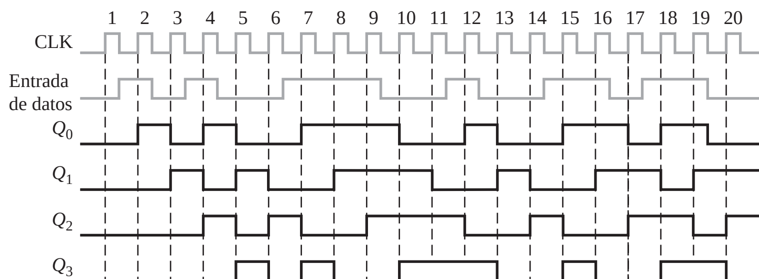


FIGURA P.73

11. *Véase* la Figura P.74.

13. *Véase* la Figura P.75.

15. *Véase* la Figura P.76.

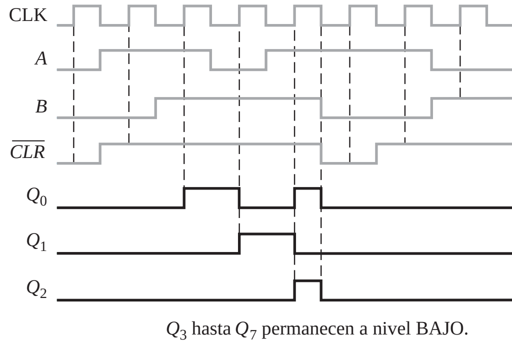


FIGURA P.74

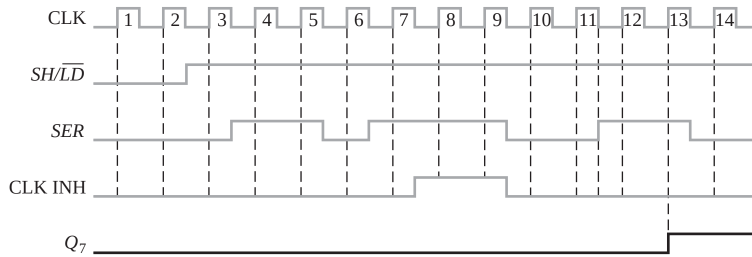


FIGURA P.75

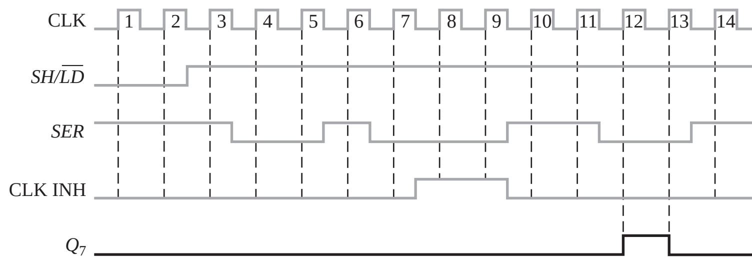


FIGURA P.76

17. *Véase* la Figura P.77.

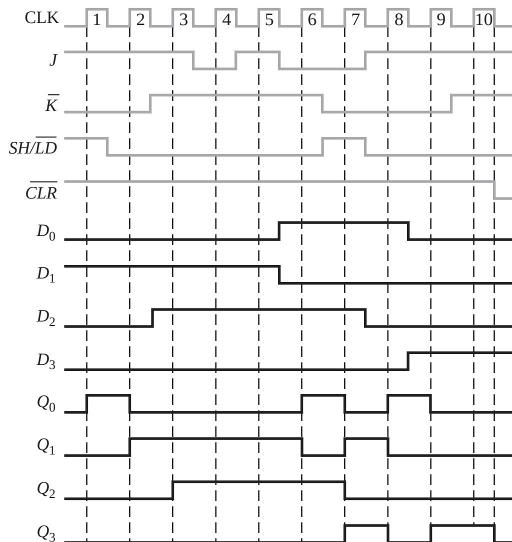


FIGURA P.77

19.

Inicialmente (76):	01001100	
CLK1:	10011000	izquierda
CLK2:	01001100	derecha
CLK3:	00100110	derecha
CLK4:	00010011	derecha
CLK5:	00100110	izquierda
CLK6:	01001100	izquierda
CLK7:	00100110	derecha
CLK8:	01001100	izquierda
CLK9:	00100110	derecha
CLK10:	01001100	izquierda
CLK11:	10011000	izquierda

21. *Véase* la Figura P.78.

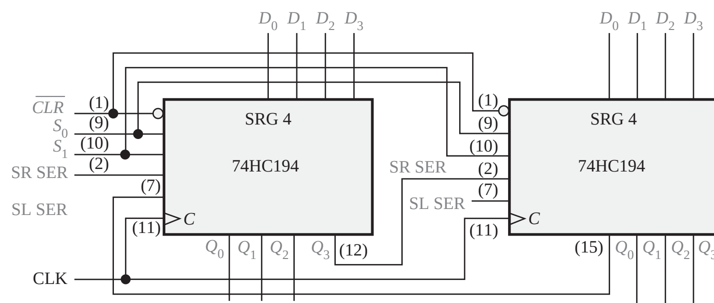


FIGURA P.78

23. **(a)** 3 **(b)** 5 **(c)** 7 **(d)** 8

25. *Véase* la Figura P.79.

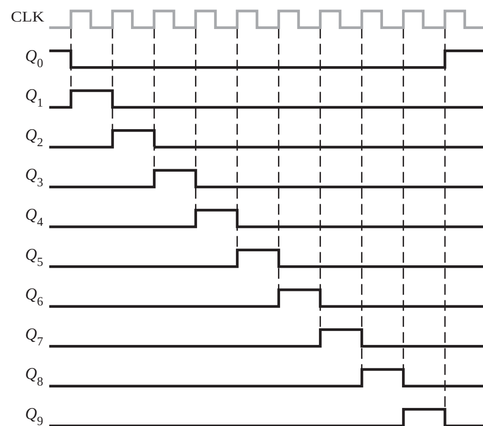


FIGURA P.79

27. *Véase* la Figura P.80.

29. Se puede producir un código incorrecto

31. La entrada D_3 está en circuito abierto.

33. **(a)** No se produce un impulso de reloj cuando se cierra el interruptor porque falla la puerta NAND (negativa-OR) o el monoestable. La entrada de reloj (*C*) del registro del código tecleado está en circuito abierto. La entrada $SH/\overline{LD}$ del registro del código tecleado está en circuito abierto.

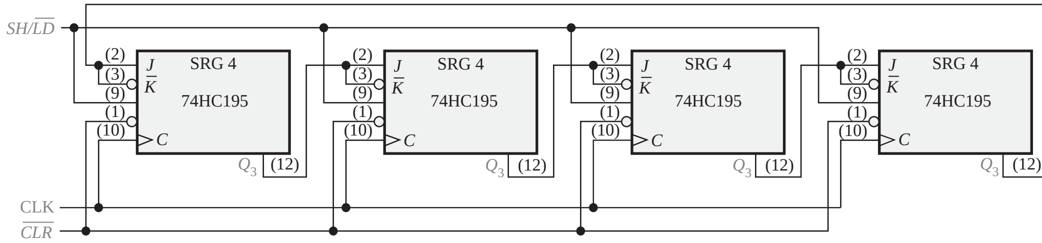


FIGURA P.80

(b) El diodo de la tercera fila está en circuito abierto. La salida Q_2 del contador en anillo está en circuito abierto.

(c) La entrada de la puerta NAND (negativa-OR) conectada a la primera columna está en circuito abierto o cortocircuitada.

(d) La entrada “2” del decodificador de columnas está en circuito abierto.

35. **(a)** El contenido del registro de salida permanece constante.

(b) El contenido de ambos registros no cambia

(c) La salida de la tercera etapa del registro de salida de datos permanece a nivel ALTO.

(d) El generador de reloj se desactiva después de cada impulso por lo que el flip-flop pasa continuamente del estado SET al estado RESET.

37. Registro de desplazamiento A: 1001

Registro de desplazamiento C: 00000100

39. *Flip-flop* de control: 7476

Generador de reloj: 555

Contador: 74LS163

Registro de entrada de datos: 74LS164

Registro de salida de datos: 74LS199

Monoestable: 74121

41. *Véase* la Figura P.81.

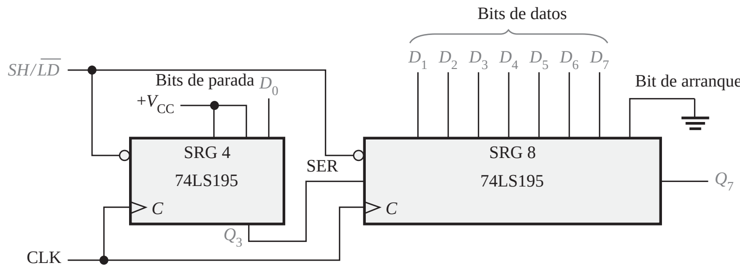


FIGURA P.81

43. Véase la Figura P.82.

CAPÍTULO 10

1. **(a)** ROM **(b)** RAM

3. El *bus de direcciones* facilita la transferencia de los códigos de dirección a la memoria para acceder a cualquier posición de memoria en cualquier orden para efectuar una operación de lectura o escritura. El *bus de datos* facilita la transferencia de los datos entre el microprocesador y la memoria o la E/S.

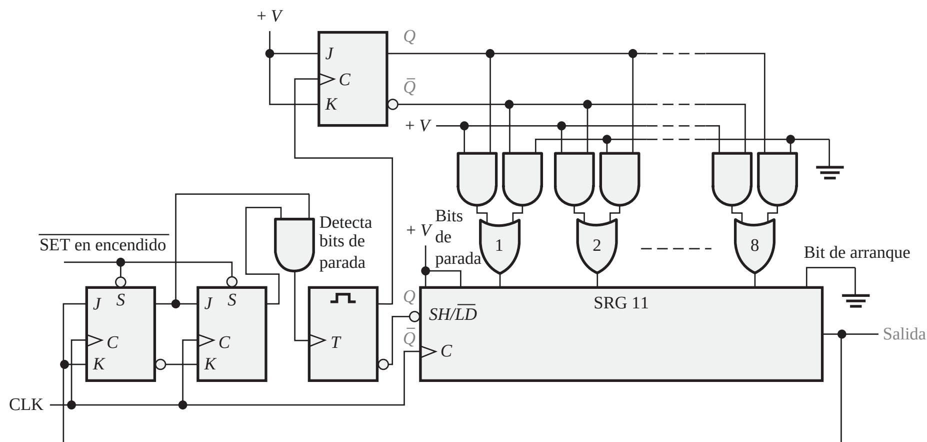


FIGURA P.82

5.

	Bit 0	Bit 1	Bit 2	Bit 3
Fila 0	1	0	0	0
Fila 1	0	0	0	0
Fila 2	0	0	1	0
Fila 3	0	0	0	0

7. 512 filas × 128 columnas de 8 bits.

9. Una SRAM almacena bits en flip-flops de forma indefinida mientras que esté aplicada la alimentación. Una DRAM almacena bits en condensadores que deben refrescarse periódicamente para conservar los datos.

11. *Véase* la Tabla P.11.

Entradas A_1	A_0	**Salidas** S_3	S_2	S_1	S_0
0	0	0	1	0	1
0	1	1	0	0	1
1	0	1	1	1	0
1	1	0	0	1	0

TABLA P.11

13. *Véase* la Figura P.83.

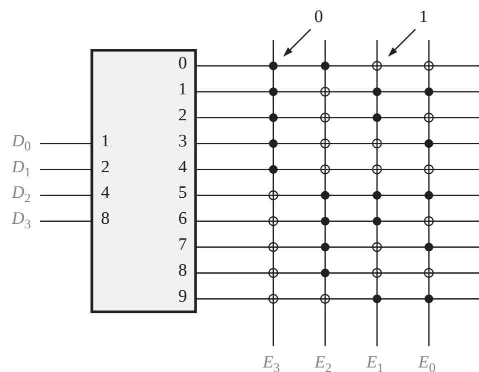


FIGURA P.83

15. Hilos fusibles:1-17, 19-23, 25-31, 34, 37, 38, 40-47, 53, 55, 58, 61, 62, 63, 65, 67, 69.

17. Utilizar ocho memorias RAM de 16k × 4 con dieciséis líneas de dirección. Dos de las líneas de dirección se decodifican para habilitar los chips de memoria seleccionados. Cuatro líneas de datos se usan para cada chip.

19. 8 bits, 64k palabras; 4 bits, 256k palabras.

21. Dirección más baja: $FC0_{16}$

Dirección más alta: $FFF_{16.}$

23. Un disco duro se formatea en pistas y sectores. Cada pista se divide en una serie de sectores, teniendo cada sector de una pista una dirección física. Normalmente, los discos duros tienen desde unos pocos cientos de pistas hasta unos pocos miles.

25. La cinta magnética tiene un tiempo de acceso mayor que el disco porque debe accederse a los datos de forma secuencial en lugar de forma aleatoria.

27. La suma de comprobación contiene un error.

29. **(a)** ROM 2 **(b)** ROM 1

(c) Todas las memorias ROM.

31. 10

33. Una PROM mantiene el código cuando no se alimenta. El código de la PROM no puede cambiarse, a menos que se trate de una EEPROM.

35. Para acomodar un código de entrada de 5 bits, el registro de desplazamiento C debe cargarse con cinco 0s en lugar de con cuatro. El nivel ALTO (1) debe desplazarse hacia la izquierda un lugar en las entradas paralelo.

CAPÍTULO 11

1. $X = \bar{A}\bar{B}\bar{C} + \bar{A}B\bar{C} + A\bar{B}C$

3. **(a)** El dispositivo PAL16L2 es una PAL (*Programmable Array Logic*, dispositivo lógico de matriz programable) con 16 entradas y dos salidas activas a nivel BAJO (L).

(b) El dispositivo PAL12H6 es una PAL (*Programmable Array Logic*, dispositivo lógico de matriz programable) con 12 entradas y 6 salidas activas a nivel ALTO (H).

5. Un CPLD está formado básicamente por múltiples SPLD que pueden ser conectados con una matriz de interconexiones programable.

7. **(a)** $\bar{A}BC\bar{D}$ **(b)** $ABC(\bar{D} + \bar{E}) = ABC\bar{D} + ABC\bar{E}$

9. $X = A\bar{B} + \bar{A}B$

11. $X_1 = A\bar{B}C\bar{D} + \bar{A}BCD + ABC\bar{D}$

$X_2 = ABCD + AB\bar{C}D + \bar{A}BC\bar{D} + A\bar{B}CD$

13. **(a)** Combinacional; 1 **(b)** Registrada; 0

15. **(a)** Registrada **(b)** GCK1 **(c)** 0 **(d)** 0

17. Salida suma de productos $= \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C + \bar{A}BC + A\bar{B}C + AB\bar{C}$

19. Bloques LUT para la lógica combinacional, la lógica del sumador y la lógica de registro.

21. *Véase* la Figura P.84.

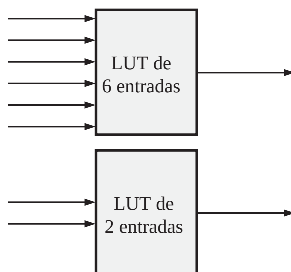


FIGURA P.84

23. *Véase* la Figura P.85.

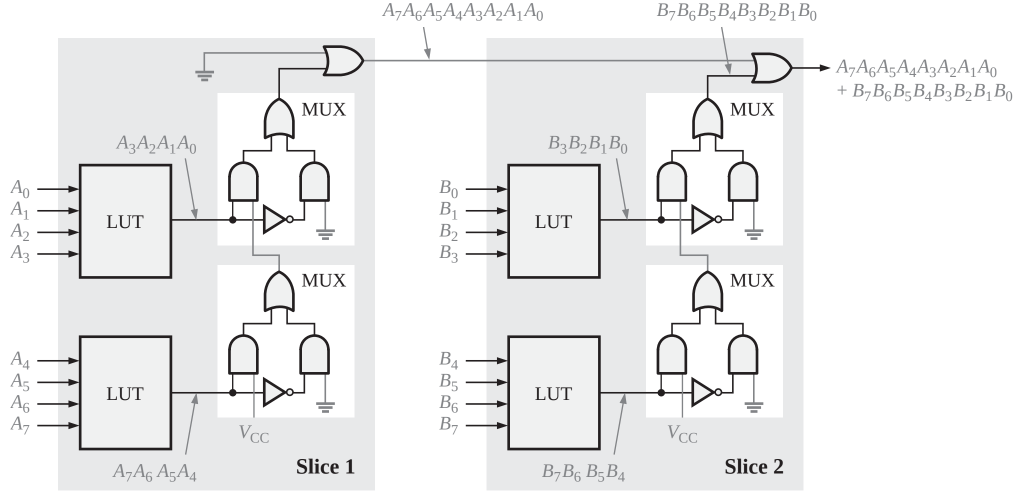


FIGURA P.85

25. Un *slice*.

27. *Véase* la Figura P.86.

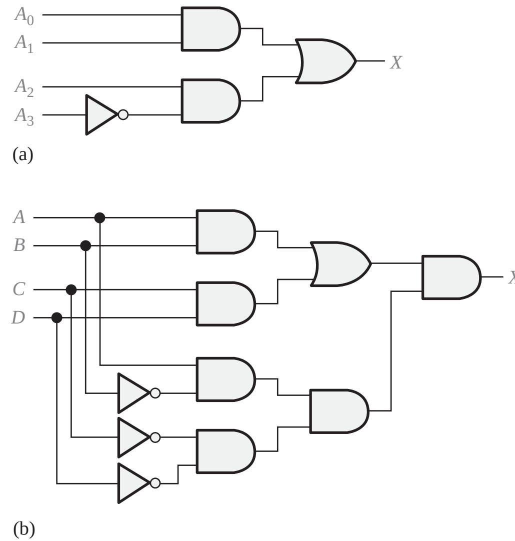


FIGURA P.86

29. *Véase* la Figura P.87 en la página siguiente.

31. Entrada Shift = 1, los datos se aplican a SDI, pasan a través del multiplexor MUX y se introducen en el registro de captura A con el flanco anterior del impulso de reloj. Desde la salida del registro de captura A, los datos pasan a través del multiplexor superior y se introducen en el registro de captura B con el flanco posterior del impulso de reloj.

33. PDI/O = 0 y OE = 0. Los datos se aplican al pin de entrada y pasan a través del multiplexor seleccionado hasta la lógica programable interna.

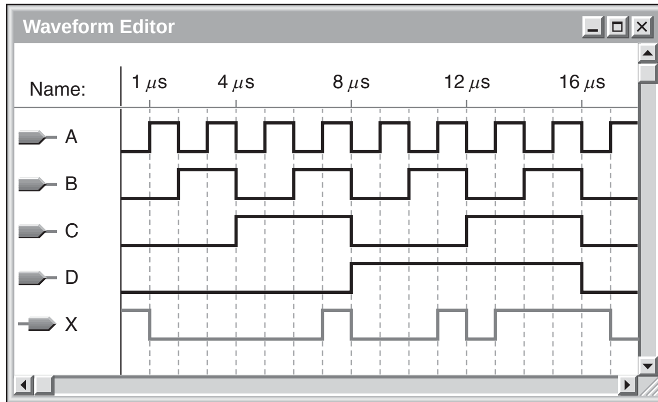


FIGURA P.87

35. 000011001010001111011

0	**0000**11001010001111011
1	0**0001**1001010001111011
3	00**0011**001010001111011
6	000**0110**01010001111011
12	0000**1100**1010001111011
9	00001**1001**010001111011
2	000011**0010**10001111011
5	0000110**0101**0001111011
10	00001100**1010**001111011
4	000011001**0100**01111011
8	0000110010**1000**1111011
1	00001100101**0001**111011
3	000011001010**0011**11011
7	0000110010100**0111**1011
15	00001100101000**1111**011
14	000011001010001**1110**11
13	0000110010100011**1101**1
11	00001100101000111**1011**

37. La entrada D a la lógica falla o no está conectada.

CAPÍTULO 12

1. CPU, memoria, puertos de E/S, buses.

3. Un bus es un conjunto de conexiones y especificaciones eléctricas que permiten trasladar información en una computadora.

5. UAL, decodificador de instrucciones, matriz de registros y unidad de control.

7. Bus de direcciones, bus de datos y bus de control

9. Carga, decodificación, ejecución.

11. CS, DS, SS, ED, FS, GS.

13. AH y AL son registros de 8 bits y representan la parte superior e inferior del registro AX de 16 bits. El EAX es un registro de 32 bits que incluye el registro AX que contiene los 16 bits inferiores.

15. Permite que dos instrucciones se ejecuten al mismo tiempo.

17. *Véase* la Figura P.88

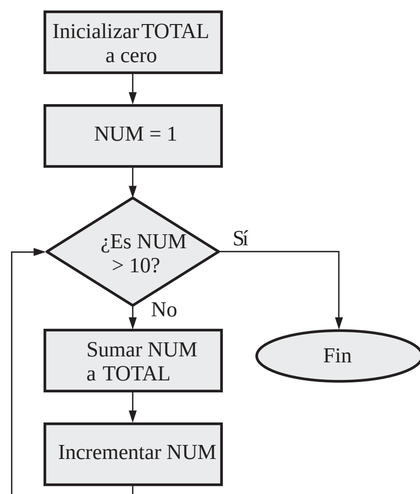


FIGURA P.88

19. Cuando se ejecuta la instrucción **mov ax, [bx]**, la palabra en memoria a la que apunta el registro bx se copia en el registro ax.

21. En una E/S por sondeo, la CPU sondea a cada uno de los dispositivos por turnos para ver si necesita que le proporcione servicio; en un sistema dirigido por interrupciones, el dispositivo periférico indica a la CPU cuándo necesite que le dé servicio.

23. Una instrucción de programa que invoca a una rutina de servicio de interrupción.

25. El mecanismo de acceso directo a memoria (DMA) puentea a la CPU.

27. *Véase* la Figura P.89.

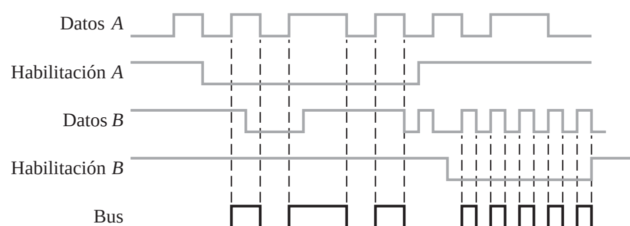


FIGURA P.89

29. El bus local es la colección de buses que establecen la interfaz directamente con el procesador. El bus PCI se utiliza para los dispositivo de expansión y se conecta al bus local a través de un controlador de bus.

31. El bus PCI es un bus de expansión a 33 o 66 MHz de 32 o 66 bits. El bus ISA es un bus de expansión a 8,33 MHz de 8 o 16 bits.

33. DCE (*Data Communication Equipment*, equipo de comunicación de datos), como por ejemplo un módem. DTE (*Data Terminal Equipment*, equipo de terminal de datos), como por ejemplo una computadora. Ambos acrónimos están asociados con el estándar RS-232/EIA-232.

35. Seis

37. Un controlador está enviando datos a dos receptores (escuchas). El primero de los dos bytes de datos (3F y 41) llega al receptor con la dirección 001A. El segundo de los dos bytes llega al receptor con la dirección 001B. Las señales de negociación ($\overline{\text{DAV}}$, $\overline{\text{NFRD}}$ y $\overline{\text{NDAC}}$) indican que la transferencia de datos se ha llevado a cabo con éxito. *Véase* la Figura P.90.

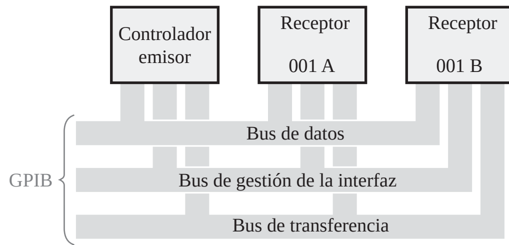


FIGURA P.90

CAPÍTULO 13

1. Un convertidor analógico-digital convierte una señal analógica en un código digital.

3. Un convertidor digital-analógico transforma un código digital en la correspondiente señal analógica.

5. *Véase* la Figura P.91.

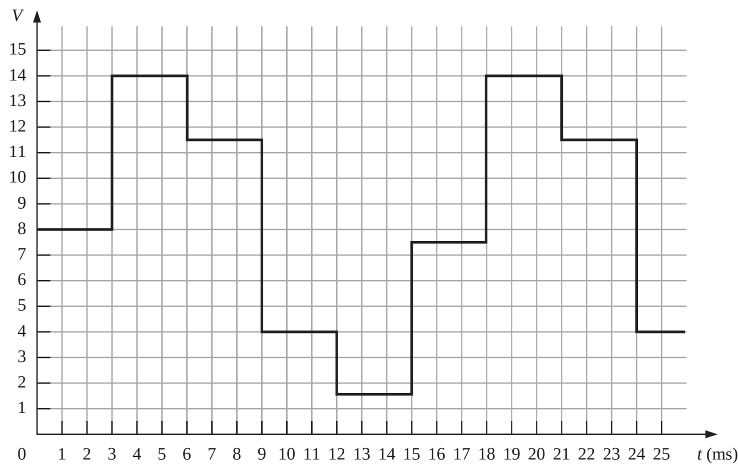


FIGURA P.91

7. 1000, 1110, 1011, 0100, 0001, 0111, 1110, 1011, 0100.

9. *Véase* la Figura P.92.

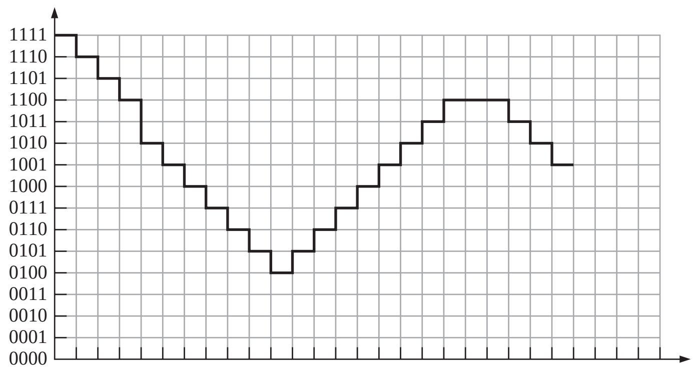


FIGURA P.92

11. 330 kΩ

13. 000, 001, 100, 110, 101, 100, 011, 010, 001, 001, 011, 110, 111, 111, 111, 111, 111, 111, 111, 100

15. *Véase* la Tabla P.12.

17. 8000 MB/s

SAR	Comentario
1000	Mayor que V_{in}, reinicializa el MSB
0100	Menor que V_{in}, mantiene el estado de entrada
0110	Igual a V_{in}, mantiene el estado final

TABLA P.12

19. 1.000.000.000

21. Despacho de instrucciones (DP): los paquetes de la instrucción se dividen en paquetes de ejecución y se asignan a las unidades funcionales. Decodificación de instrucciones (DC): las instrucciones se decodifican.

23. *Véase* la Figura P.93.

25. **(a)** 14,3% **(b)** 0,098% **(c)** 0,00038%

27. *Véase* la Figura P.94.

FIGURA P.93

FIGURA P.94

CAPÍTULO 14

1. No, $V_{OH(mín)} < V_{IH(mín)}$

3. 0,15 V en el estado ALTO; 0,25 V en el estado BAJO.

5. Puerta *C*

7. 16 ns

9. Puerta *C*

11. Si, G_2

13. **(a)** activado **(b)** bloqueado
(c) bloqueado **(d)** activado

15. Consulte la Figura P.95 para ver un posible circuito.

17. **(a)** ALTO (H) **(b)** flotante
(c) ALTO (H) **(d)** alta-*Z*

19. **(a)** BAJO (L) **(b)** BAJO (L) **(c)** BAJO (L)

21. *Véase* la Figura P.96.

23. **(a)** R_p = 198 Ω
(b) R_p = 198 Ω
(c) R_p = 198 Ω

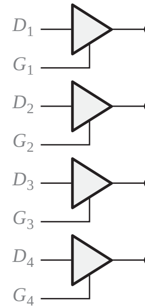


FIGURA P.95

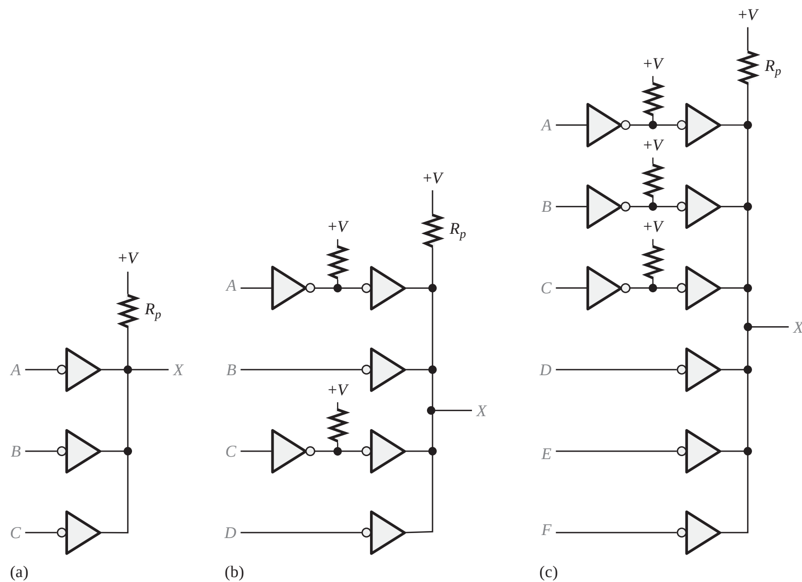


FIGURA P.96

25. ALVC

27. **(a)** *A, B* a *X*: 9,9 ns
C, D a *X*: 6,6 ns

(b) *A* a X_1, X_2, X_3: 14 ns
B a X_1: 7 ns
C a X_2: 7 ns
D a X_3; 7 ns

(c) *A* a *X*: 11,1 ns
B a *X*: 11,1 ns
C a *X*: 7,4 ns
D a *X*: 7,4 ns

29. ECL opera con transistores BJT no saturados

GLOSARIO

Acarreo Dígito generado cuando la suma de dos dígitos binarios excede en 1.

Acarreo anticipado Método de suma binaria en que los acarreos de las etapas sumadoras precedentes se anticipan, eliminando de esta manera los retardos en la propagación de los acarreos.

Acarreo serie Método de suma binaria en el que el acarreo de salida de cada sumador se convierte en el acarreo de entrada del sumador siguiente.

Aceptor Dispositivo receptor en un bus.

ACSII *American Standard Code for Information Interchange*, código estándar americano para el intercambio de información; el código alfanumérico más utilizado.

ADC flash o paralelo Convertidor analógico-digital simultáneo.

Adyacencia Característica de las celdas en un mapa de Kamaugh en el que sólo se cambia una variable de una celda a otra inmediata a ella por cualquiera de sus cuatro lados.

Aestable Que no tiene ningún estado estable. Un multivibrador aestable oscila entre dos estados semi-estables.

Alfanumérico Que contiene números, letras y otros caracteres.

Álgebra booleana Las matemáticas de los circuitos lógicos.

Aliasing El efecto creado cuando se muestrea una señal a menos de dos veces la frecuencia de señal máxima. El *aliasing* crea frecuencias no deseadas que interfieren con la frecuencia de la señal.

Almacenamiento Capacidad de los dispositivos digitales para guardar bits. Proceso de conservación de datos para su uso posterior.

Alta-Z Estado de alta impedancia de un circuito triestado, en el que la salida se desconecta de forma efectiva del resto del circuito.

ALU (*Arithmetic Logic Unit*) Unidad aritmético lógica; el elemento clave de procesamiento de un microprocesador, que realiza operaciones aritméticas y lógicas.

Amplificador operacional Dispositivo con dos entradas diferenciales que tiene una ganancia muy alta, muy alta impedancia de entrada y muy baja impedancia de salida.

Amplitud En un tren de impulsos, la altura o máximo valor del impulso medido desde su nivel más bajo.

Analógico Que es continuo o toma valores continuos, en oposición a tomar valores discretos.

Anchura del impulso (t_W) Intervalo de tiempo entre los puntos del 50% del valor de los flancos anterior y posterior de un impulso. La duración de un pulso.

AND Operación lógica básica en la que se obtiene una salida verdadera (nivel ALTO) sólo si todas las condiciones de entrada son verdaderas (nivel ALTO).

ANSI *American National Standards Institute*, Instituto Nacional Americano de Normalización.

Antifusible Tipo de conexión programable no volátil de un PLD que se puede dejar abierta o se puede cortocircuitar una sola vez según indique el programa.

Arbitraje del bus El proceso que evita que dos orígenes utilicen un bus al mismo tiempo.

Arquitectura La unidad VHDL que describe el funcionamiento interno de una función lógica. La disposición funcional interna de los elementos, que dota a un dispositivo de características de operación particulares.

Asíncrono Que no tiene ninguna relación temporal fija. Que no ocurre simultáneamente.

Basculación Acción de un flip-flop cuando cambia de estado con cada impulso de reloj.

Base Una de las tres regiones de un transistor bipolar.

BCD *Binary Coded Decimal*, código decimal binario, código digital en el que cada dígito decimal, de 0 a 9, se representa mediante un grupo de cuatro bits.

BEDO DRAM Memoria de acceso aleatorio dinámica con salida de datos extendida a ráfagas.

Bidireccional Que posee dos direcciones. En un registro de desplazamiento bidireccional, los datos almacenados se pueden desplazar a la derecha o a la izquierda.

Biestable Que tiene dos estados estables. Los flip-flops y los *latches* son multivibradores biestables.

Binario Que tiene dos valores o estados; describe un sistema de numeración en base dos y utiliza el 1 y el 0 como dígitos.

BIOS Sistema básico de entrada/salida. Conjunto de programas en ROM que establece la interfaz con los dispositivos de E/S en un sistema de computadora.

Bipolar Transistor que tiene dos tipos de portadores de carga opuestos dentro de su estructura.

Bit Dígito binario que puede ser 1 ó 0.

Bit de paridad Bit que se añade a cada grupo de bits de información para hacer que el número de unos sea par o impar en dicho grupo de bits.

Bit de signo El bit más a la izquierda de un número binario que designa si el número es positivo (0) o negativo (1).

Bit más significativo (MSB, *Most Significant Bit*) El bit más a la izquierda de un número entero o código binario.

Bit menos significativo (LSB, *Least Significant Bit*) Generalmente, el bit de más a la derecha de un número entero o código binario.

BIU *Bus Interface Unit*, unidad de interfaz de bus. La parte de la CPU que establece la interfaz con los buses del sistema y extrae las instrucciones, lee los operandos y escribe los resultados.

BJT *Bipolar Junction Transistor*, transistor bipolar de unión; dispositivo semiconductor utilizado para conmutación o amplificación. Un BJT tiene dos uniones, la unión base-emisor y la unión base-colector.

Borrado (*clear*) Entrada asíncrona utilizada para resetear un flip-flop (pone la salida Q a 0). Cuando se pone un registro o contador en el estado en que contiene solamente ceros.

Buffer Circuito que evita la carga de una entrada o salida.

Bus Conjunto de interconexiones que establece la interfaz entre uno o más dispositivos basándose en una especificación estandarizada.

Bus de control Un conjunto de conductores de una sola dirección que conecta la CPU con otras partes de la computadora, para coordinar sus operaciones y comunicarse con dispositivos externos.

Bus de datos Un conjunto bidireccional de conductores por los que los datos o códigos de instrucciones se transfieren al microprocesador o por los que se envía, desde el microprocesador, el resultado de una operación o cálculo.

Bus de direcciones Un grupo de conductores de una sola dirección que va desde el microprocesador a la memoria, u otro dispositivo externo, por el que se envía el código de dirección.

Bus IEEE-488 También denominado GPIB (bus de interfaz de propósito general). Bus paralelo estándar utilizado ampliamente como interfaz de prueba y medida.

Bus IEEE-1349 Bus serie de transferencia de datos a alta velocidad. También conocido como *FireWire*.

Bus ISA *Industry Standard Architecture*, arquitectura estándar de la industria. Estándar de bus paralelo interno.

Bus local Bus interno que conecta el microprocesador a la memoria caché, la memoria principal, el coprocesador y el controlador de bus PCI.

Bus PCI *Peripheral Control Interconnect*, bus de interconexión de control de periféricos. Un estándar de bus paralelo interno.

Byte Grupo de ocho bits.

Cadena Una secuencia contigua de bytes o palabras.

Cama de pinchos Un método para la prueba automatizada de tarjetas de circuito, según el cual la tarjeta se monta sobre un utillaje que recuerda una cama de pinchos y que hace contacto con los puntos de prueba.

Capacidad de palabra El número de palabras que una memoria puede almacenar.

Capacidad Número total de unidades de datos (bits, *nibbles*, bytes, palabras) que puede almacenar una memoria.

Carácter Símbolo, letra o número.

Carga (*load*) Introducir datos en un registro de desplazamiento.

Carga unidad Una medida del *fan-out*. Una entrada de puerta representa una carga unidad para la salida de la puerta, dentro de la misma familia de circuitos integrados.

Cartucho Jaz Un dispositivo de almacenamiento. Discos duros encerrados en un cartucho de plástico rígido, con capacidades de almacenamiento de 1 Gbyte o 2 Gbytes.

Cascada Conectar dispositivos “uno tras otro”, como cuando se conectan varios contadores de forma que la salida de un contador esté conectada a la entrada de habilitación del siguiente contador.

CCD *Charge-Coupled Device*, dispositivo de acoplamiento de carga; un tipo de memoria semiconductora que almacena datos en forma de paquetes de carga y a la que se accede en serie.

CD-R CD grabable. Dispositivo de almacenamiento de disco óptico en el que los datos se pueden almacenar una vez.

CD-ROM Dispositivo de almacenamiento en disco óptico en el que se prealmacenan los datos y sólo se pueden leer.

CD-RW CD-reescribible. Disco óptico en el que se pueden escribir y sobrescribir muchas veces los datos.

Celda Una zona de un mapa de Karnaugh que representa una única combinación de variables en forma de producto. Un elemento de almacenamiento en una memoria. Un punto de cruce de una fila y una columna en un PLD. Un elemento de almacenamiento de una memoria.

Ciclo de trabajo La relación entre la anchura de un impulso y el período, expresado en porcentaje.

Circuito Disposición de componentes eléctricos y/o electrónicos interconectados de manera que realicen una función específica.

Circuito integrado (CI) Un tipo de circuito en el que todos sus componentes se encuentran integrados en un único chip semiconductor de muy pequeño tamaño.

Circuito secuencial Circuito digital cuyos estados lógicos dependen de una determinada secuencia temporal.

CLB *Configurable Logic Block*, bloque lógico configurable. Una unidad de lógica en una FPGA que está compuesta de múltiples bloques lógicos más pequeños y de una interconexión programable local que se emplea para conectar entre sí los módulos lógicos que componen el CLB.

CMOS *Complementary Metal-Oxide Semiconductor*, metal-óxido semiconductor complementario. Un tipo de circuito de transistores que se implementa con un tipo de transistores de efecto de campo.

Cociente El resultado de una división.

Codec Un codificador y decodificador combinado.

Codificador Circuito digital (dispositivo) que convierte la información a un formato codificado.

Codificador de prioridad Codificador en el que sólo se codifica el dígito de entrada de valor más alto, ignorándose cualquier otra entrada activa.

Código Un conjunto de bits ordenados según un patrón único y utilizados para representar información tal como números, letras y otros símbolos. En VHDL, instrucciones de programa.

Código de operación Código que representa una instrucción particular del microprocesador. Un mnemónico.

Código Gray Código digital sin pesos caracterizado por el cambio de un único bit entre números codificados adyacentes en una secuencia.

Código Hamming Un tipo de código de corrección de errores.

Código máquina Instrucciones básicas en código binario que el procesador es capaz de comprender.

Código reubicable Programa que puede desplazarse a cualquier posición dentro del espacio de memoria sin tener que modificar el código básico.

Cola Memoria de alta velocidad que almacena instrucciones o datos.

Colector Una de las tres regiones de un transistor bipolar.

Colector abierto Un tipo de salida en los circuitos TTL en el que el colector del transistor de salida se deja desconectado internamente y se encuentra disponible para conectarse a una carga externa que requiera una corriente o tensión relativamente altas.

Comparador Circuito digital que compara las magnitudes de dos cantidades y produce una salida que indica la relación entre estas cantidades.

Compilador Programa de aplicación disponible en paquetes de desarrollo que controla el flujo de diseño y traduce el código fuente en código objeto en un formato que puede ser lógicamente probado o descargado en un dispositivo de destino.

Complemento El inverso u opuesto de un número. En el álgebra booleana es la función inversa, que se expresa mediante una barra por encima de la variable. El complemento de 1s 0 y viceversa.

Condición de carrera Condición en una red lógica en la que la diferencia entre los tiempos de propagación a través de dos o más caminos de señal en la red puede dar lugar a una salida incorrecta.

Conexión en cascada Conectar la salida de un dispositivo a la entrada de un dispositivo similar, permitiendo a uno de los dispositivos excitar a otro, para aumentar la capacidad de operación.

Contador Circuito digital capaz de contar sucesos electrónicos, tales como impulsos, avanzando a través de una secuencia de estados binarios.

Contador ascendente/descendente Contador que puede avanzar tanto en una dirección como en otra a lo largo de una secuencia.

Contador asíncrono Un tipo de contador en el que se utiliza como entrada de reloj de cada etapa la salida de la etapa precedente.

Contador de décadas Contador digital de diez estados.

Contador de propagación Un contador asíncrono.

Contador en anillo Registro en el que un determinado patrón de 1s y 0s circula continuamente.

Contador Johnson Un tipo de registro en el que se desplaza un patrón pre-almacenado de 1s y 0s a través de las etapas, creando una secuencia patrón única de bits.

Contador síncrono Un tipo de contador en el que todas las etapas utilizan el mismo impulso de reloj.

Contienda de bus Una condición adversa que podría ocurrir si dos o más dispositivo intentan comunicarse a un mismo tiempo a través de un bus.

Contiguo Unido.

Controlador Instrumento que puede definir a los demás instrumentos conectados al bus como transmisores o receptores durante una transferencia de datos.

Conversión analógico-digital (A/D) Proceso de convertir una señal analógica en digital.

Conversión digital-analógica (D/A) El proceso de convertir una secuencia de códigos digitales a formato analógico.

Convertidor analógico-digital (ADC) Dispositivo utilizado para convertir una señal analógica en una secuencia de códigos digitales.

Convertidor digital-analógico (DAC) Dispositivo utilizado para convertir una información en formato digital en un formato analógico.

CPLD Un dispositivo programable lógico complejo que está compuesto de múltiples matrices SPLD con interconexiones programables.

CPU *Central processing Unit*, unidad central de proceso. Uno de los componentes principales de todas las computadoras que controla el funcionamiento interno y procesa los datos. El módulo de un DSP que procesa las instrucciones del programa.

Cronograma *Véase* diagrama de tiempos.

Cuantificación El proceso mediante el que se asigna un código binario a cada valor muestreado durante la conversión analógico digital.

Diagrama de estados Representación gráfica de una secuencia de estados o valores.

Diagrama de tiempos Gráfico de formas de onda digitales que muestra la relación temporal existente entre todas las señales y cómo varía cada una respecto a las restantes.

DAT *Digital Audio Tape*, cinta digital de audio. Un tipo de formato de cinta magnética.

Datos Información en formato numérico, alfabético o cualquier otro.

DCE *Data Communications Equipment*, equipo de comunicación de datos.

Debug Programa incluido en el sistema operativo DOS, que permite realizar diversas operaciones sobre archivos e incluye un programa ensamblador rudimentario. Permite eliminar un problema de hardware o software.

Década Que se caracteriza por diez estados o valores.

Decimal Describe un sistema de numeración en base diez.

Decodificación Una etapa de las operaciones *pipeline* de un DSP en la que las instrucciones se asignan a unidades funcionales y se decodifican.

Decodificador Circuito digital (dispositivo) que convierte la información codificada en un formato más familiar o no codificado.

Decrementar Disminuir el estado binario de un contador en una unidad.

Demultiplexor (DEMUX) Circuito (dispositivo digital) que conmuta los datos digitales desde una línea de entrada a varias líneas de salida según una secuencia temporal especificada.

Desbordamiento La condición que se produce cuando el número de bits en una suma excede el número de bits de cada uno de los números que se suman.

Descarga Proceso del flujo de diseño en el que el diseño lógico se transfiere desde el software al hardware.

Desplazamiento La distancia en número de bytes de una dirección física desde la dirección base.

Desplazar Mover datos binarios de una etapa a otra dentro de un registro de desplazamiento o de otro dispositivo de almacenamiento; mover datos binarios dentro o fuera de un dispositivo.

Detección de errores Proceso de detección de los bits erróneos de un código digital.

Diagrama de estados Una descripción gráfica de una secuencia de estados o valores.

Diferencia El resultado de una sustracción.

Digital Relacionado con los dígitos o con cantidades discretas; que posee un conjunto de valores discretos, en oposición a tener valores continuos.

Dígito Símbolo utilizado para expresar una cantidad.

DIMM Módulo de memoria de terminal doble.

Diodo Dispositivo semiconductor que permite el paso de corriente sólo en una dirección.

Dirección Posición de una determinada celda de almacenamiento o grupo de celdas en memoria. Posición de memoria diferenciada, que contiene un byte.

Dirección base La dirección inicial de un segmento de memoria.

Dirección física Posición real de una unidad de datos en la memoria.

Disco duro Dispositivo de almacenamiento magnético; normalmente, una pila de dos o más discos rígidos encerrados en un compartimento sellado.

Disco magneto-óptico Dispositivo de almacenamiento que utiliza medios electromagnéticos y un haz láser para leer y escribir datos.

Disco Zip Un tipo de dispositivo de almacenamiento magnético. Disco flexible con una capacidad de 100 MB contenido en un cartucho de plástico rígido de aproximadamente el tamaño de un disquete.

Disipación de potencia El producto de la tensión de alimentación continua y la corriente de alimentación continua en un circuito electrónico. Cantidad de potencia requerida por un circuito.

Disparador Impulso que se utiliza para iniciar un cambio en el estado de un circuito lógico.

Dispositivo de destino Dispositivo PLD montado sobre un útil de programación o una tarjeta de desarrollo en la que se descarga un diseño lógico software. El dispositivo lógico programable que se quiere programar.

Disquete o disco flexible Dispositivo de almacenamiento magnético. Típicamente suele ser un disco flexible de 3,5 pulgadas, con una capacidad de almacenamiento de 1,44 MB, ubicado en el interior de una funda de plástico rígido.

Dividendo En una división, la cantidad que está siendo dividida.

Divisor En una operación de división, la cantidad por la que se divide el dividendo.

DLT *Digital Linear Tape* (cinta digital lineal). Un tipo de formato de cinta magnética.

DMA *Direct Memory Access*, acceso directo a memoria. Método para conectar de forma directa un dispositivo periférico con la memoria, sin utilizar la CPU como mecanismo de control.

Dominio Todas las variables utilizadas en una expresión booleana.

DRAM *Dynamic Random-Access Memory*, memoria dinámica de acceso aleatorio. Un tipo de memoria semiconductora que utiliza condensadores como elemento de almacenamiento y es una memoria de lectura/escritura volátil.

Drenador Unos de los terminales de un transistor de efecto de campo.

DSP *Digital Signal Processsor*, procesador digital de señal; un tipo especial de microprocesador que procesa los datos en tiempo real.

DTE *Data Terminal Equipment*, equipo terminal de datos.

Duración del bit Intervalo de tiempo ocupado por un único bit dentro de una secuencia de bits; el período del reloj.

DVD-ROM *Digital Versatil Disk*-ROM (disco digital versátil ROM). También conocido como disco digital de vídeo-ROM. Un tipo de dispositivo de almacenamiento óptico en el que los datos se prealmacenan y que tiene una capacidad mucho mayor que un CD-ROM.

E^2CMOS *Electrically Erasable* CMOS (EECMOS), CMOS eléctricamente borrable. Tecnología de circuitos utilizada para las celdas reprogramables de un dispositivo PLD.

ECL *Emitter-Coupled Logic*, lógica de emisor acoplado; un tipo de circuito lógico integrado que se implementa con transistores bipolares no saturados.

EDIF *Electronic Design Interchange Format*, formato de intercambio de diseños electrónicos. Un formato estándar de lista de interconexiones (*netlist*).

EDO DRAM Memoria de acceso aleatorio dinámica con salida de datos extendida.

EEPROM *Electrically Erasable Programmable Read-Only Memory*, memoria programable de sólo lectura eléctricamente borrable. Tipo de conexión programable no volátil de un PLD basado en celdas de memoria de sólo lectura eléctricamente programables y borrables, que se pueden activar y desactivar repetidamente por programación.

Ejecución Proceso de la CPU durante el cual se lleva a cabo una instrucción. Una etapa de las operaciones *pipeline* de un DSP en la que se llevan a cabo las instrucciones decodificadas.

Elemento lógico La sección más pequeña de la lógica de una FPGA, que habitualmente contiene una tabla LUT, lógica asociada y un flip-flop.

Emisor Una de las tres regiones de un transistor bipolar.

Emparejamiento de instrucciones El proceso de combinar ciertas instrucciones independientes, de modo que puedan ser ejecutadas simultáneamente por dos unidades de ejecución separadas.

Encapsulado DIP (*Dual In-line Package*) Un tipo de encapsulado de CI cuyos terminales deben insertarse en los taladros de la tarjeta de circuito impreso.

Ensamblador Programa que convierte los mnemónicos en código máquina.

Ensamblador cruzado Un programa que traduce un programa en lenguaje ensamblador para un tipo de microprocesador en un lenguaje de ensamblador para otro tipo de microprocesador.

Entero Un número entero.

Entidad Unidad de VHDL que describe las entradas y las salidas de una función lógica.

Entrada La señal o línea que entra en un circuito. Señal que controla el funcionamiento de un circuito.

Entrada/Salida (E/S) Un terminal de un dispositivo que puede ser utilizado como entrada o como salida.

EPROM *Erasable Programmable Read-Only Memory*, memoria de sólo lectura programable y borrable. Tipo de conexión programable no volátil de un PLD basado en celdas de memoria de sólo lectura eléctricamente programables y que se pueden activar y desactivar repetidamente por programación.

Escritura El proceso de almacenamiento de datos en memoria.

Establecimiento de la comunicación, negociación (*handshaking*) Método o proceso de intercambio de señales, mediante el que dos dispositivos o sistemas digitales establecen conjuntamente una comunicación.

Etapa Elemento de almacenamiento (flip-flop) en un registro.

Exploración de contorno (*boundary scan*) Un método para probar internamente un dispositivo PLD y que está basado en el estándar JTAG (IEEE Std. 1149.1).

Exponente Parte de un número en coma flotante que representa el número de lugares que la coma decimal (o la coma binaria) tiene que moverse.

Expresión booleana Ordenación de variables y operadores lógicos utilizada para expresar el funcionamiento de un circuito lógico.

Extracción Un proceso de la CPU en el que se obtiene una instrucción de la memoria. Una etapa de las operaciones *pipeline* de un DSP en la que se obtiene una instrucción de la memoria de programa.

Fan–out Número de entradas de puertas equivalentes de la misma familia que puede excitar una puerta lógica.

FET *Field-Effect Transistor*, transistor de efecto de campo.

FIFO *First In-First Out*, primero en entrar-primero en salir.

FireWire El bus serie estándar IEEE-1394.

Flanco anterior La primera transición de un impulso.

Flanco posterior La segunda transición de un impulso.

Flip-flop Circuito básico de almacenamiento que puede almacenar sólo un bit a un tiempo; dispositivo biestable síncrono.

Flip-flop D Un tipo de multivibrador biestable en el que la salida sigue al estado de la entrada D en el flanco de disparo de la señal de reloj.

Flip-flop disparado por flanco Un tipo de flip-flop en el que los datos se introducen y aparecen en la salida durante el mismo flanco del impulso del reloj.

Flip-flop J-K Un tipo de flip-flop que puede funcionar en los modos de SET, RESET, no cambio y basculación.

Flip-flop S-R Flip-flop SET-RESET.

Flujo de diseño El proceso de secuencia de operaciones llevado a cabo para programar un dispositivo de destino.

FPGA *Field Programmable Gate Array*, matriz de puertas programable sobre el terreno: un dispositivo lógico programable que utiliza una carga LUT como elemento lógico básico y que emplea, generalmente, tecnología de proceso basada en antifusible o basada en SRAM.

FPM DRAM Memoria de acceso aleatorio dinámica en modo página rápido.

Frecuencia (*f*) El número de impulsos por segundo que hay en una onda periódica. La unidad de frecuencia es el hertzio.

Frecuencia de Nyquist La frecuencia más alta de señal que puede muestrearse a una determinada frecuencia de muestreo; se trata de una frecuencia igual o inferior a la mitad de la frecuencia de muestreo.

Fuente Uno de los terminales de un transistor de efecto de campo.

Fuente de corriente Funcionamiento de un circuito cuando su salida entrega corriente a la carga.

Fusible Tipo de conexión programable no volátil de un PLD que se puede dejar cortocircuitada o se puede dejar en abierto una sola vez según indique el programa

GAL *Generic Array Logic*, dispositivo lógico de matriz genérica. Un tipo reprogramable de dispositivo SPLD, que es similar a una PAL salvo porque utiliza una tecnología de proceso reprogramable, como por ejemplo celdas EEPROM (E^2CMOS) en lugar de fusibles.

Generación de acarreo El proceso mediante el que se produce un acarreo de salida en un sumador completo cuando las dos entradas son 1.

Glitch Pico de tensión o de corriente de corta duración, no deseado y, generalmente, producido de forma no intencionada.

GPIB *General Purpose Interface Bu*s, bus de interfaz de propósito general, basado en el estándar IEEE 488.

Habilitar Activar o poner en modo operacional. Una entrada de un circuito lógico que activa su funcionamiento.

Hardware La circuitería y componentes físicos de un sistema de computadora (en oposición a las instrucciones que denominamos software).

HDL *Hardware Description Language*, lenguaje de descripción hardware. Lenguaje utilizado para describir un diseño lógico utilizando software.

Herramienta de encaje Una herramienta del software de compilación que selecciona las interconexiones óptimas, las asignaciones óptimas y las asignaciones óptimas de celdas lógicas con el fin de encajar un diseño dentro del dispositivo de destino seleccionado.

Hexadecimal Describe un sistema de numeración en base 16.

Histéresis Una característica de los circuitos con umbral de disparo, como el *trigger* Schmitt, donde el dispositivo se activa y desactiva para diferentes niveles de entrada.

Hoja de especificaciones Documento que especifica los valores de los parámetros y las condiciones de funcionamiento de un circuito integrado o de otro dispositivo.

HPIB *Hewlett-Packard Interface Bus*, bus de interfaz de Hewlett-Packard; igual que el GPIB (bus de interfaz de propósito general).

I^2L *Integrated Injection Logic*, lógica de inyección integrada; un tipo de tecnología de circuitos integrados.

IEEE *Institute of Electrical and Electronic Engineers.*

Implementación Proceso software en el que las estructuras software descritas por la lista de interconexiones se implementan en la estructura del dispositivo de destino.

Impulso Cambio brusco desde un nivel a otro, seguido, tras un cierto intervalo de tiempo, por otro cambio brusco que lo devuelve a su nivel original.

Incrementar Aumentar el estado binario de un contador en una unidad.

Indicador Bit que indica el resultado de una operación aritmética o lógica, o que se emplea para alterar una operación.

Indiferente Combinación de literales de entrada que no pueden ocurrir y que se utilizan como 1s o 0s en un mapa de Karnaugh, con propósitos de simplificación.

Inicialización (preset) Entrada asíncrona para inicializar un flip-flop (pone la salida Q a 1).

Inicio de un nuevo ciclo Sufrir una transición (como ocurre en los contadores) desde el estado final o terminal hasta el estado inicial.

Inmunidad al ruido Capacidad de un circuito para rechazar señales no deseadas.

Instrucción Un paso de un programa de computadora. Unidad de información que le dice a la CPU lo que tiene que hacer.

Interconexión local Un conjunto de líneas que permite establecer interconexiones entre los ocho elementos lógicos de un bloque LAB sin emplear interconexiones de fila y columna.

Interfaz Mecanismo por el que los dos o más dispositivos electrónicos o sistemas se hacen compatibles operacionalmente entre sí, de manera que puedan funcionar adecuadamente juntos.

Interrupción Señal o instrucción que hace que el proceso actual sea temporalmente detenido mientras se ejecuta una rutina de servicio.

Interrupción software Una instrucción que invoca una rutina de servicio de interrupción.

Introducción de texto Un método de introducir un diseño lógico dentro del software utilizando un lenguaje de descripción hardware (HDL, *Hardware Description Language*).

Introducción de esquemáticos Un método de introducir un diseño lógico dentro del software utilizando símbolos esquemáticos.

Inversión La conversión de un nivel ALTO (1) a un nivel BAJO (0), o viceversa.

Inversor Circuito NOT. Circuito que cambia un nivel ALTO en un nivel BAJO, y viceversa.

IP Puntero de instrucción. Un registro especial de la CPU que almacena el desplazamiento correspondiente a la siguiente instrucción que se va a ejecutar.

ISP *In-system Programming*, programación dentro del sistema: un método para programar dispositivos SPLD después de haber sido instalados en una tarjeta de circuito impreso y están funcionando en el sistema.

LAB *Logic Array Block*, bloque de matriz lógica. Un grupo de macroceldas que se pueden interconectar con otros bloques LAB o con otras E/S utilizando una matriz de interconexiones programables. También se denomina bloque funcional.

Latch Dispositivo digital biestable utilizado para almacenar un bit.

LCCC *Leadless Ceramic Chip Carrier*, encapsulado SMT que tiene contactos metálicos moldeados en su cuerpo.

LCD *Liquid Crystal Display*, display de cristal líquido.

Lectura El proceso de recuperar datos de una memoria.

LED *Light-Emitting Diode*, diodo emisor de luz.

Lenguaje de alto nivel Un tipo de lenguaje de computadora muy próximo al lenguaje humano que se encuentra en un nivel por encima del lenguaje ensamblador.

Lenguaje ensamblador Un lenguaje de programación que utiliza palabras similares a las del idioma inglés y que tiene una correspondencia biunívoca con el lenguaje máquina.

Lenguaje máquina Instrucciones de computadora escritas en código binario que una computadora es capaz de comprender. El nivel más bajo de lenguaje de programación.

Ley asociativa En la suma (operación OR) y multiplicación (operación AND) de tres o más variables, el orden en que se agrupan las variables no altera el resultado.

Ley conmutativa En la suma (OR) o multiplicación (AND) de dos variables, el orden en el que las variables se suman o multiplican no altera el resultado.

Ley distributiva Ley que dice que si sumamos (operación OR) varias variables y luego multiplicamos (operación AND) el resultado por una única variable, lo que nos queda es equivalente a multiplicar (AND) la variable aislada por cada una de las otras variables y luego sumar (OR) todos los términos.

LIFO *Last In-First Out*, memoria de tipo primero en entrar-último en salir. Pila de memoria.

Lista de interconexiones (*netlist*) Un listado detallado de información necesaria para describir un circuito, como por ejemplo tipos de elementos, entradas y salidas, y todas las interconexiones.

Literal Una variable o el complemento de una variable.

Localización de averías Técnica que consiste en identificar, aislar y corregir sistemáticamente cualquier fallo en un circuito o sistema.

Lógica combinacional Combinación de puertas lógicas interconectadas para producir una determinada función booleana sin capacidad de almacenamiento o de memoria. En ocasiones se denomina *lógica combinatoria.*

Lógica En la electrónica digital, la capacidad de los circuitos de puertas de tomar decisiones, en los que un nivel ALTO representa una proposición verdadera y un nivel BAJO representa una proposición falsa.

Lógica positiva Sistema que representa un 1 binario con un nivel ALTO y un 0 binario con un nivel BAJO.

Longitud de palabra El número de bits en una palabra.

LSI *Large-Scale Integration*, integración a gran escala; un nivel en la complejidad de los CI de función fija que tienen entre 100 y 10.000 puertas equivalente por chip.

LUT *Look-Up Table*, tabla de consulta: un tipo de memoria que puede programarse para generar funciones suma de productos.

Macrocelda Una matriz lógica de suma de productos con salidas combinacional y registrada. Parte de una PAL, GAL o CPLD, que está compuesta, generalmente, por una puerta OR y cierta lógica de salida asociada. Múltiples macroceldas interconectadas forman un CPLD.

Magnitud El tamaño o valor de una cantidad.

Mantisa El módulo de un número en coma flotante.

Mapa de Karnaugh Disposición de celdas que representa las combinaciones de literales en una expresión booleana y que se utiliza para la simplificación sistemática de la expresión.

Máquina de estados Sistema lógico que exhibe una secuencia de estados condicionada por la lógica interna y las entradas externas. Cualquier circuito secuencial que exhibe una determinada secuencia de estados.

Margen de ruido La diferencia entre la salida máxima a nivel BAJO de una puerta y el máximo nivel de entrada a nivel BAJO aceptable por una puerta equivalente; también, la diferencia entre la salida mínima a nivel ALTO de una puerta y el mínimo nivel de entrada a nivel ALTO aceptable por una puerta equivalente.

Matriz En un PLD, una matriz formada por filas de términos productos y columnas de líneas de entrada, con una celda programable en cada unión. En VHDL, una matriz es un conjunto ordenado de elementos individuales que tienen un mismo nombre de identificador.

Matriz AND Una matriz de puertas AND que consta de una matriz de interconexiones programable.

Matriz de memoria Matriz formada por las celdas de memoria colocadas formando filas y columnas.

Matriz de registros Un conjunto de posiciones de almacenamiento temporal dentro del microprocesador, para almacenar datos y direcciones a las que tiene que acceder rápidamente el programa.

Memoria caché Memoria de alta velocidad, relativamente pequeña, que almacena las instrucciones o datos más recientemente utilizados procedentes de la memoria principal, más grande pero más lenta.

Memoria dinámica Memoria semiconductora que tiene celdas de almacenamiento capacitivas que tienden a perder los datos que llevan almacenados más de un cierto período de tiempo y, por tanto, tienen que ser refrescadas.

Memoria estática Memoria semiconductora volátil que utiliza flip-flops como celdas de almacenamiento y que es capaz de mantener los datos indefinidamente sin necesidad de refresco.

Memoria flash Memoria semiconductora de acceso aleatorio de lectura/escritura no volátil, en la que los datos se almacenan como carga en la puerta flotante de un determinado tipo de FET.

MFLOPS *Million Floating-Point Operations Per Second*, millones de operaciones en coma flotante por segundo.

Microprocesador Circuito integrado digital que puede ser programado mediante una serie de instrucciones para realizar funciones específicas sobre los datos.

Minimización Proceso por el que se obtiene una expresión en forma de suma de productos o de producto de sumas, conteniendo el menor número posible de términos con el menor número posible de literales por término.

Minuendo Número del que se sustrae otra cantidad.

MIPS *Million Instructions Per Second*, millones de instrucciones por segundo.

MMACS *Million Multiply/Accumulates Per Second*, millones de operaciones de multiplicación/acumulación por segundo.

Mnemónico Instrucción similar al idioma inglés que es convertida por un programa ensamblador en código máquina para que lo use un procesador.

Módem Modulador/demodulador que sirve para conectar dispositivos digitales con sistemas de transmisión analógicos, como las líneas telefónicas.

Modo real Modo de operación de un procesador Intel que emula la memoria de 1 Mbyte del 8086.

Modulación delta Un método de conversión analógico-digital utilizando un proceso de cuantización de 1 bit.

Módulo El número de estados en la secuencia de un contador.

Módulo DSP La unidad central de proceso de un DSP.

Módulo hardware Parte fija de la de una FPGA que implementa el fabricante con el fin de proporcionar una función específica.

Módulo software Una parte de la lógica de una FPGA. Similar al módulo hardware excepto porque tiene algunas características programables.

Monoestable Que tiene un solo estado estable. Un multivibrador monoestable, o sencillamente monoestable, produce un único impulso en respuesta a una entrada de disparo.

Monotonicidad Característica de un DAC definida por la ausencia de inversiones de paso incorrectas. Un tipo de linealidad digital-analógica.

MOS *Metal-Oxide Semiconductor*, semiconductor de metal-óxido. Un tipo de tecnología de transistores.

MOSFET *Metal-Oxide Semiconductor Field Effect Transistor*, transistor de efecto campo de semiconductor metal-óxido.

MSI *Medium-Scale Integration*, integración a media escala; un nivel en la complejidad de los CI que tienen entre 10 y 100 puertas equivalentes por chip.

Muestreo El proceso de tomar un número suficiente de valores discretos en determinados punto de una forma de onda, definiéndose con esos valores dicha forma de onda.

Multiplexor (MUX) Circuito (dispositivo digital) que conmuta los datos digitales de distintas líneas de entrada a una única línea de salida según una secuencia temporal especificada.

Multiplicación booleana En el álgebra booleana, la operación AND.

Multiplicador El número que multiplica al multiplicando.

Multiplicando El número que está siendo multiplicado por otro número.

Multivibrador Una clase de circuitos digitales en los que la salida se conecta de vuelta a la entrada (lo cual se denomina realimentación) para producir dos estados estables, un único estado estable o ningún estado estable, dependiendo de la configuración.

Negativa-AND Operación equivalente a una puerta NOR, en la que se obtiene una salida a nivel ALTO cuando todas sus entradas están a nivel BAJO.

Negativa-OR Operación equivalente a una puerta NAND, en la que se obtiene una salida a nivel ALTO cuando una o más entradas están a nivel BAJO.

Nibble Un grupo de cuatro bits.

NMOS *n-channel Metal-Oxide Semiconductor*, semiconductor de metal-óxido de canal *n*.

No volátil Término que describe la memoria que puede mantener los datos almacenados cuando se desconecta la alimentación.

Nodo Punto de conexión común en un circuito, en el que la salida de una puerta se conecta a una o más entradas de puerta.

NOT Operación lógica básica que realiza inversiones.

Notación de dependencia Sistema de notación para los símbolos lógicos que especifica las relaciones de entrada y salida, definiendo una determinada función de forma completa; parte integral de un estándar ANSI/IEEE 91-1984.

Numérico Relativo a los números.

Número en coma flotante Representación de un número basada en la notación científica, en la que el número consta de un exponente y una mantisa.

Octal Describe un sistema de numeración en base ocho.

OLMC *Output-Logic Macrocell*, macrocelda lógica de salida. Parte de una GAL que puede programarse para obtener salidas combinacional y registrada. Un bloque de lógica en un dispositivo GAL que contiene una puerta OR fija y otra lógica para manipular las entradas y/o las salidas.

OR Una operación lógica básica en la que una salida verdadera (nivel ALTO) se produce cuando una o más de las condiciones de entrada son verdaderas (nivel ALTO).

OR-exclusiva (XOR) Operación lógica elemental en la que ocurre un nivel ALTO cuando las dos entradas tienen niveles opuestos.

Origen Dispositivo que envía algo por un bus.

Oscilador Circuito electrónico que está basado en el principio de la realimentación regenerativa y que produce una señal de salida repetitiva. Fuente de señal.

OTP *On-time Programmable*, programable una sola vez.

PAL *Programmable Array Logic*, dispositivo lógico de matriz programable. Un tipo de dispositivo CPLD programable una única vez que está compuesta por una matriz programable de puertas AND que se conecta a una matriz fija de puertas OR.

Palabra Unidad completa de datos binarios.

Paralelo En los circuitos digitales, datos que se producen simultáneamente a través de varias líneas. Transferencia o procesamiento de varios bits simultáneamente.

Paridad En relación a los códigos binarios, tener un número par o impar de unos en un grupo de código.

Paridad impar Condición de poseer un número impar de 1s en cada grupo de bits.

Paridad par Condición por la que se tiene un número par de 1s en cada grupo de bits.

Periférico Dispositivo o instrumento que proporciona servicios de comunicación con alguna computadora, o proporciona servicios o funciones auxiliares a una computadora.

Periódico Describe una señal que se repite a intervalos de tiempo fijos.

Período (*T*) Tiempo requerido por las señales periódicas para repetirse.

Período de latencia El tiempo que tarda el sector deseado en colocarse bajo el cabezal, una vez que el cabezal está posicionado sobre la pista deseada de un disco duro magnético.

Peso El valor de un dígito en un número en función de su posición dentro del número.

PIC *Programmable Interrupt Controller*, controlador de interrupciones programable; maneja las instrucciones de acuerdo con su prioridad.

Pipeline Cuando se aplica a las memorias, una implementación que permite que se inicie una operación de lectura o de escritura antes de que la operación anterior se complete. Parte de la arquitectura de un DSP que permite procesar múltiples instrucciones.

PLA *Programmable Logic Array*, matriz lógica programable. Un dispositivo SPLD con matrices AND y OR programables.

PLCC *Plastic Leaded Chip Carrier*, encapsulado STM cuyos terminales se insertan en su cuerpo en forma de J.

PLD *Programmable Logic Device*, dispositivo lógico programable. Circuito integrado que puede programarse para realizar cualquier función lógica especificada.

PMOS *p-channel Metal-Oxide Semiconductor*, semiconductor de metal-óxido de canal *p*.

Polarización directa Tensión de polarización que permite a una unión *pn* semiconductora en un transistor o diodo conducir corriente.

Polarización inversa Condición de polaridad del voltaje que evita que una unión *pn* de un transistor o diodo pueda conducir corriente.

Precarga El proceso de ejecutar instrucciones al mismo tiempo que otras instrucciones son "cargadas", eliminando tiempo de inactividad. Este proceso también se denomina *pipeline.*

Primitiva Un elemento lógico básico tal como una puerta, un flip-flop, un pin de entrada/salida, una conexión de masa o una conexión V_{CC}.

Producto El resultado de una multiplicación.

Producto de sumas Expresión booleana que consiste simplemente en multiplicar (operación AND) términos suma (operación OR).

Producto velocidad-potencia Parámetro de funcionamiento que consiste en el producto del retardo de propagación y la disipación de potencia en un circuito digital.

Programa Lista de instrucciones de computadora organizado para llegar a un resultado específico. Software.

Programa fuente Un programa escrito en lenguaje ensamblador o en un lenguaje de alto nivel.

Programa objeto Traducción a lenguaje máquina de un programa fuente en un lenguaje de alto nivel.

PROM *Programmable Read-Only Memory*, un tipo de memoria semiconductora programable de sólo lectura. Un SPLD con una matriz AND fija y una matriz OR programable; se utiliza como dispositivo de memoria, pero, generalmente, no como circuito lógico.

Propagación de acarreo Proceso mediante el que se propaga el acarreo de entrada de un sumador completo hasta que se convierte en acarreo de salida, cuando uno o los dos bits de entrada son 1 y el acarreo de entrada es 1.

Propiedad intelectual (IP) Diseños que son propiedad de un fabricante de dispositivos lógicos programables.

Pseudo-operación Instrucción dirigida al programa ensamblador (en oposición a las instrucciones dirigidas a un procesador).

Puerta Circuito lógico que realiza una operación lógica específica, tal como AND u OR. Uno de los tres terminales de un transistor de efecto de campo.

Puerta AND Puerta lógica que produce una salida a nivel ALTO sólo cuando todas las entradas están a nivel ALTO.

Puerta NAND Puerta lógica que produce una salida a nivel BAJO sólo si todas las entradas están a nivel ALTO.

Puerta NOR Puerta lógica en la que la salida es un nivel BAJO cuando al menos una de las entradas está a nivel ALTO.

Puerta NOR-exclusiva (XNOR) Puerta lógica que produce una salida a nivel BAJO sólo cuando las dos entradas tienen niveles opuestos.

Puerta OR Puerta lógica que produce una salida a nivel ALTO cuando una o más entradas están a nivel ALTO.

Puerta OR-exclusiva (XOR) Puerta lógica que produce una salida a nivel ALTO sólo cuando las dos entradas tienen niveles opuestos.

Puerta universal Tanto una puerta NAND como NOR. El término *universal* se refiere a la propiedad de aquellas puertas que permiten que cualquier operación lógica pueda ser implementada mediante ellas o mediante una combinación de puertas de ese tipo.

Puerto Una interfaz física en una computadora, a través de la cual se transfieren datos hacia o desde un periférico.

Puerto de E/S Puerto de entrada/salida. La interfaz entre un bus interno y un periférico.

Puntero El contenido de un registro (o registros) que almacena una dirección.

QIC *Quarter-Inch Cassette.* Un tipo de cinta magnética.

RAM *Random-Access Memory*, memoria de acceso aleatorio. Memorias semiconductoras volátiles de lectura/escritura.

Realimentación Tensión de salida o porción de ésta que se conecta de vuelta a la entrada de un circuito.

Receptor Instrumento capaz de recibir datos en un bus GPIB (*General-Purpose Interface Bus*, bus de interfaz de propósito general).

Refresco Renovar los contenidos de una memoria dinámica, recargando las celdas de almacenamiento de tipo capacitivo.

Registrada Configuración de salida de una macrocelda de un CPLD, en la que la salida procede de un flip-flop.

Registro Circuito digital capaz de almacenar y desplazar información binaria; típicamente utilizado como dispositivo de almacenamiento temporal.

Registro de desplazamiento universal Registro que tiene capacidad de entradas y salidas serie y paralelo.

Reloj Señal de temporización básica de un sistema digital. Forma de onda periódica en la que el intervalo entre impulsos es igual a la duración de un bit. La entrada de disparo de un flip-flop.

RESET Estado de un flip-flop o *latch* cuando la salida es 0. La acción de producir un estado de RESET.

Resistencia de *pull-up* Resistencia que se utiliza para mantener un determinado punto de un circuito a nivel ALTO cuando se encuentra en estado inactivo.

Resolución El número de bits utilizado en un convertidor ADC.

Restador Circuito lógico utilizado para sustraer dos números binarios.

Resto Cantidad sobrante tras una división.

Retardo de propagación Intervalo de tiempo que transcurre entre la transición de entrada y su correspondiente transición de salida en un circuito lógico.

ROM *Read-Only Memory*, memoria semiconductora no volátil de acceso aleatorio de sólo lectura. También se conoce como ROM de máscara.

Salida Señal o línea que sale de un circuito.

Schottky Un tipo de tecnología de circuitos de lógica transistor-transistor.

SCSI *Small Computer System Interface* (interfaz de sistemas informáticos de pequeño tamaño). Estándar de bus paralelo externo.

SDRAM Memoria de acceso aleatorio dinámica síncrona.

Segmento Un bloque de memoria de 64k.

Seguimiento de señales Técnica de localización de averías mediante la cual se observan las señales paso a paso, comenzando en la entrada y siguiéndolas hasta la salida, o viceversa. En cada punto, las formas de ondas observadas se comparan con la señal correcta que debería haber en ese punto.

Selector de datos Circuito que selecciona los datos de varias entradas a un tiempo según una secuencia y los coloca en la salida; también se denomina multiplexor.

Semisumador Circuito digital que suma dos bits y produce una suma y un acarreo de salida. No puede manejar acarreos de entrada.

Señal Un tipo de objeto VHDL que almacena datos.

Serie Que tiene un elemento seguido de otro, como ocurre en una transmisión serie de bits; que se produce, como los impulsos, en secuencia en vez de simultáneamente.

SET Estado de un flip-flop o *latch* cuando la salida es 1. La acción de producir un estado SET.

SIMM *Single-In-line Memory Module*, módulo de memoria de una sola fila de terminales.

Simulación de temporización Un proceso software que utiliza la información sobre los retardos de propagación y los datos de la lista de interconexiones para comprobar tanto la operación lógica como la temporización de caso peor de un diseño.

Simulación funcional Un proceso software que comprueba la operación lógica y funcional de un diseño.

Síncrono Que tiene una relación temporal fija. Que ocurre de forma simultánea.

Síntesis Proceso software por el que un diseño se convierte en una lista de interconexiones (*netlist*).

SMT *Surface-Mount Technology*, tecnología de montaje superficial. Técnica de encapsulado de los CI en la que los encapsulados son más pequeños que los de los DIP y se montan en la superficie de una tarjeta de circuito impreso.

Software Programa informático. Programas que dicen a la computadora qué es lo que tiene que hacer para poder realizar un determinado número de tareas.

SOIC *Small-Outline Integrated Circuit*. Un encapsulado SMT que recuerda un pequeño DIP pero que tiene sus pines con forma de alas de gaviota.

Sonda Accesorio utilizado para aplicar una tensión a la entrada de un osciloscopio u otro instrumento.

Sonda volante Un método para la prueba automatizada de tarjetas de circuito mediante el cual se mueven una o más sondas de un lugar a otro con el fin de hacer contacto con los puntos de prueba.

Sondeo El proceso de comprobar una serie de dispositivos periféricos para determinar si algunos de ellos requiere servicio por parte de la CPU.

SPLD *Simple Programmable Logic Device*, dispositivo lógico programable simple. Matriz de puertas AND y puertas OR que se puede programar para obtener funciones lógicas específicas. Existen cuatro tipos de estos dispositivos: PROM, PLA, PAL y GAL.

SRAM *Static Random Access Memory*, memoria estática de acceso aleatorio. Un tipo de conexión programable volátil de dispositivos PLD basada en celdas de memoria estáticas de acceso aleatorio y que por programación se pueden activar y desactivar de forma repetida.

SSI *Small-Scale Integration*, integración a pequeña escala; un nivel en la complejidad de los CI de función fija que tienen hasta 10 puertas equivalentes por chip.

Subrutina Serie de instrucciones que pueden agruparse y que sólo se programan una vez, y que un programa puede utilizar repetidamente

Suma El resultado que se obtiene cuando se suman dos o más números.

Suma booleana En el álgebra booleana, la operación OR.

Suma de productos Expresión booleana que consiste simplemente en sumar (operación OR) términos que contienen productos (operación AND).

Sumador Circuito lógico que se emplea para sumar dos números binarios.

Sumador completo Circuito digital que suma dos bits y un acarreo de entrada para producir una suma y un acarreo de salida.

Sumando Los dos operandos que se suman en la operación de la adición.

Sumidero de corriente Funcionamiento de un circuito cuando su salida acepta corriente de la carga.

Supresión de cero Proceso de eliminar los ceros anteriores o posteriores en una presentación en display digital.

Sustraendo Número del que se resta el minuendo.

Tabla de verdad Tabla que muestra las entradas y los correspondientes niveles de salida de un circuito lógico.

Tasa de procesamiento La velocidad media con la que se ejecuta un programa.

Temporizador Circuito que puede ser utilizado como monoestable o como oscilador. Un circuito que genera una salida con un intervalo de tiempo fijo.

Término producto Producto booleano de dos o más literales equivalente a una operación AND.

Término suma Suma booleana de dos o más literales equivalente a la operación OR.

Tiempo de acceso Tiempo que transcurre desde que se aplica una dirección de memoria válida hasta que aparecen datos de salida válidos.

Tiempo de búsqueda Tiempo que tarda el cabezal de lectura-escritura de un disco duro en posicionarse sobre la pista deseada para realizar una operación de lectura.

Tiempo de caída Intervalo de tiempo que transcurre entre el punto en que la señal alcanza el 90% de su valor y el punto en que alcanza el 10% durante el flanco negativo de un impulso.

Tiempo de establecimiento (set-up) Intervalo de tiempo requerido por los niveles de control para estar en las entradas de un circuito digital, como puede ser un flip-flop, antes del flanco de disparo del impulso de reloj.

Tiempo de mantenimiento (hold) El intervalo de tiempo requerido por los niveles de control para permanecer en las entradas de un flip-flop después del flanco de disparo del reloj, de manera que se active fiablemente el dispositivo.

Totem-pole Un tipo de salida de los circuitos TTL.

Transistor Dispositivo semiconductor que tiene ganancia en tensión y/o corriente. Cuando se utiliza como dispositivo de conmutación se aproxima a un interruptor abierto o cerrado.

Transmisor Instrumento capaz de transmitir datos por un bus GPIB (bus de interfaz de propósito general).

Triestado Un tipo de salida en un circuito lógico que posee tres estados: ALTO, BAJO y alta-Z. También se conoce como tres estados.

TTSOP *Thin Shrink Small-Outline Package.*

TTL *Transistor-Transistor Logic*, lógica transistor-transistor, un tipo de circuito integrado que utiliza transistores de unión bipolares.

TVSOP *Thin Very Small-Outline Package.*

ULSI *Ultra Large-Scale Integration*, integración a ultra gran escala. Un nivel en la complejidad de los CI que tienen más de 100.000 puertas equivalentes por chip.

Unidad de control La parte del microprocesador que proporciona las señales de temporización y control para introducir y extraer datos del microprocesador y para sincronizar la ejecución de instrucciones.

Unidad de ejecución (EU) La parte de la CPU que ejecuta las instrucciones. Contiene la unidad aritmético-lógica (ALU), los registros de propósito general y los indicadores (*flag*).

USB *Universal Serial Bus* (bus serie universal). Estándar de bus serie externo.

UV EPROM ROM programable borrable por rayos ultravioleta.

Validación Proceso por el que se utiliza un pulso para muestrear la aparición de un suceso, en un determinado instante de tiempo relativo a dicho suceso.

Valor de fin de cuenta Estado final de la secuencia de un contador.

Variable Símbolo utilizado para representar una magnitud lógica que puede tener tanto un valor 1 como 0; generalmente se designa mediante una letra cursiva.

VHDL Lenguaje de descripción hardware estándar. IEEE Std. 1076-1993.

VLSI *Very Large Scale Integration*, integración a muy gran escala. Un nivel en la complejidad de los CI que tienen entre 10.000 y 100.000 puertas equivalentes por chip.

Volátil La característica de un dispositivo lógico programable que describe que se pierden los datos almacenados cuando se elimina la alimentación.

WORM *Write Once-Read Many* (una escritura, muchas lecturas). Un tipo de dispositivo de almacenamiento óptico.

ÍNDICE